VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Interaktivní prohlížeč BRDF funkcí

Interactive Visualization of BRDF Functions

Richard Zvonek

Diplomová práce

Vedoucí práce: Ing. Tomáš Fabián, Ph.D.

Ostrava, 2021

Abstrakt

Tato diplomová práce se zabývá popisem aplikace určené k vizualizaci BRDF funkcí. Aplikace je interaktivní, umožňuje vizualizovat vybrané BRDF funkce, měnit parametry a vizualizovat jednoduchou fotorealistickou scénu. Diplomová práce také zpracovává nutnou teorii pro pochopení problematiky fotorealistické počítačové grafiky.

Klíčová slova

BRDF, Fotorealistický rendering, Path tracing, Monte Carlo

Abstract

This thesis is focused on description of an application intended to visualize BRDF functions. The application is interactive, it allows to visualize selected BRDF functions, change parameters and visualize a simple photorealistic scene. This thesis also elaborates the necessary theory for understanding the issue of photorealistic computer graphics.

Keywords

BRDF, Photorealistic rendering, Path tracing, Monte Carlo

Poděkování

Rád bych na tomto místě poděkoval vedoucímu práce Ing. Tomášovi Fabiánovi, Ph.D., za pomoc se zpracováním této diplomové práce a jeho věcné připomínky.

Obsah

Seznam použitých zkratek a symbolů

AABB	–	Axis-Aligned Bounding Box
BRDF	–	Bidirectional Reflectance Distribution Function
BTDF	–	Bidirectional Transmittance Distribution Function
BSDF	–	Bidirectional Scattering Distribution Function
BSSRDF	–	Bidirectional Sub-surface Scattering Reflectance Distribution Function
BVH	–	Bounding Volume Hierarchy
GLSL	–	OpenGL Shading Language
IOR	–	Index of Refraction
MIS	–	Multiple Importance Sampling
PDF	–	Probability Density Function
SAH	–	Surface Area Heuristic

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Jednou z oblastí počítačové grafiky je fotorealistická vizualizace. Aby byla vizualizace fyzikálně přesná a uvěřitelná, je potřeba přesně simulovat osvětlení objektu ve scéně. Jedna z dílčích částí jsou matematické funkce, popisující odraz světla od povrchu – BRDF funkce. Pro lepší představu chování jednotlivých funkcí je vhodné tyto funkce nějakým způsobem vizualizovat.

Teoretická část této diplomové práce se zabývá teorií nutnou pro pochopení problematiky BRDF funkcí a následným popisem aplikace, která slouží k vizualizaci BRDF funkcí. Aplikace, která je pak implementována v praktické části této práce může být primárně využita ke vzdělávacím účelům – k demonstraci různých BRDF funkcí, k experimentování s jejich parametry, nebo také k návrhu materiálů dále použitých v aplikacích pro tvorbu fotorealistických vizualizací.

Aplikace umožňuje zobrazení jednotlivých BRDF funkcí. Jednotlivé funkce lze interaktivně ovlivňovat pomocí změny parametrů, náhled výsledného fotorealistického materiálu je zároveň zobrazen. Aplikace má také možnost zobrazit vzorkovací funkce, které jsou pro použití s BRDF funkcemi důležité.

V první kapitole je sjednoceno značení, které je použito ve vzorcích této práce. Druhá kapitola je věnována rešerši, jsou popsány existující aplikace zabývající se vizualizací BRDF funkcí. Popsány jsou jejich funkce, výhody a nedostatky. Na konci druhé kapitoly jsou specifikovány konkrétní požadavky na implementovanou aplikaci. Ve třetí kapitole je čtenář seznámen s teoretickými základy realistického osvětlení v počítačové grafice. Čtvrtá kapitola se věnuje teorii, týkající se BRDF funkcí. Dále jsou ve čtvrté kapitole zpracovány některé konkrétní BRDF funkce, které jsou následně implementovány v aplikaci. Pátá kapitola je věnována praktickému popisu implementované aplikace. Rozebrána jsou technická řešení a jednotlivé funkce aplikace. V šesté kapitole jsou teoreticky popsány optimalizační techniky používané ve fotorealistické počítačové grafice.

1.1 Užité symboly

Z důvodu použití vyššího množství matematických symbolů v této diplomové práci jsem sjednotil značení použité ve vzorcích. Přehled těchto vzorců s případným doplněním detailů je uveden v následujícím seznamu.

1.1.1 Obecné

p: Bod na povrchu tělesa

$L_e(p, \omega_i)$: Zář aktivně vyzařovaná z objektu v daném bodě

$L_r(p, \omega_o)$: Zář pasivně odražená objektem v daném bodě

$L_o(p, \omega_o)$: Výsledná zář vyzařovaná z daného bodu

$L_i(p, \omega_i)$: Zář dopadající na daný bod p z daného směru ω_i

$E(p)$: Globální ozáření daného bodu

$dE(p, \omega_i)$: Diferenciální ozáření daného bodu z daného směru ω_i

$r(p, \omega_o)$: Ray casting funkce

$f_r(p, \omega_o, \omega_i)$: 4D BRDF funkce

(θ, ϕ): Směrový vektor definovaný sférickými souřadnicemi[1]

(x, y, z): Vektor definovaný kartézskými souřadnicemi

$\hat{n}$: Vektor normály povrchu

$\hat{m}$: Vektor normály mikroplošky

ω_i: Vektor z daného bodu p ve směru ke světelnému zdroji. Platí: $\omega_i = (\theta_i, \phi_i)$

ω_o: Vektor z daného bodu p ve směru k pozorovateli. Platí: $\omega_o = (\theta_o, \phi_o)$

ω_r: Vektor zrcadlového odrazu světla, platí: $\omega_r = 2(\omega_i \cdot \hat{n})\,\hat{n} - \omega_i$

$\hat{h}$: „Poloviční“ (halfway vector) vektor mezi pohledovým a světelným vektorem. Platí: $\hat{h} = \frac{\omega_i + \omega_o}{\|\omega_i + \omega_o\|}$, $\theta_{\omega_i \hat{h}} = \theta_{\omega_o \hat{h}} = \frac{\theta_{\omega_i \omega_o}}{2}$

θ_r: Úhel odrazu, platí: $\cos\theta_r = \omega_o \cdot \omega_r$

[1] Bez informace o velikosti vektoru, v této práci předpokládám jednotkový vektor

1.1.2 Vzorkování

ξ_1: Náhodné číslo s uniformní distribucí z reálného intervalu $\langle 0, 1)$

ξ_2: Náhodné číslo s uniformní distribucí z reálného intervalu $\langle 0, 1)$

1.1.3 Parametry BRDF funkcí

ρ: Albedo materiálu, poměr mezi odraženým a pohlceným světlem (Lambert, Oren-Nayar)

k_s: Koeficient zrcadlové odrazivosti materiálu (Phong)

k_d: Koeficient difuzní odrazivosti materiálu (Phong)

n: Koeficient lesklosti materiálu (Phong)

σ: Koeficient drsnosti povrchu – roughness (Torrance-Sparrow, Cook-Torrance, Oren-Nayar)

F_0: Koeficient zrcadlové odrazivosti materiálu při nulovém incidentním úhlu (kolmý dopad světelného paprsku na povrch). Platí: $F_0 = \left(\frac{\eta_1 - \eta_2}{\eta_1 + \eta_2}\right)^2$ (Torrance-Sparrow, Cook-Torrance)

η_1: Index lomu (IOR hodnota) materiálu, na který dopadá světlo (Torrance-Sparrow, Cook-Torrance)

η_2: Index lomu (IOR hodnota) materiálu, ze kterého dopadá světlo (Torrance-Sparrow, Cook-Torrance)

Kapitola 2

Přehled existujících řešení

Existuje množství aplikací řešících podobnou tematiku. V zásadě lze tyto aplikace rozdělit do dvou kategorií:

1. Aplikace pro zobrazení BRDF funkcí

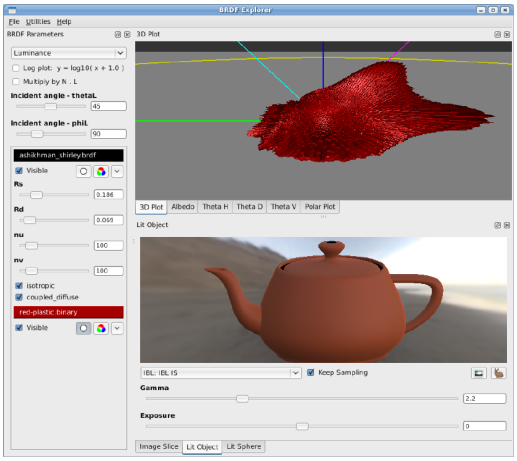
2. Aplikace pro návrh BRDF funkcí

Aplikace vytvořená v této práci spadá do první kategorie, primárně se zabývá pouze zobrazováním. V následujících odstavcích jsou popsány vybrané aplikace z obou kategorií.

2.1 Aplikace pro zobrazení BRDF funkcí

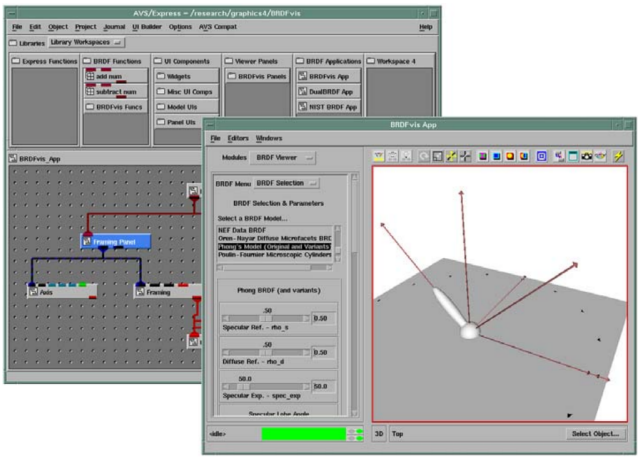
Ve článku [1] byla představena aplikace Disney BRDF Explorer [2]. Aplikace umí načítat a srovnávat analyticky zadané BRDF modely. Jednotlivé modely jsou definovány kódem pro GLSL shader, pomocí interního formátu je podporováno zadání parametrů pro jednotlivé BRDF funkce, které je poté zobrazeno v uživatelském rozhraní. Aplikaci je možné jednoduše rozšířit o uživatelsky zadané BRDF funkce při dodržení zadaného formátu. Aplikace také umí načítat a zobrazovat BRDF modely popsané pomocí měření (např. z MERL[1] databáze). Jednotlivé BRDF funkce je možné mezi sebou srovnávat a přepínat mezi nimi. Vizualizace je interaktivní, lze měnit parametry BRDF funkcí. Pohled na vizualizaci je však statický, není možné pohybovat s kamerou. Kromě pouhé vizualizace BRDF funkce je také možné zobrazit finální render s použitím zvolené BRDF funkce. Osvětlení objektu je možné pomocí bodového světla, nebo pomocí HDR mapy. Podporováno je načítání modelů, ve výchozím stavu je zobrazen model koule. Aplikace je licencována jako open-source, je multiplatformní a používá pro svou funkcionalitu knihovnu QT. Pro vykreslování je použita technologie OpenGL. Ukázka aplikace je zobrazena na obrázku 2.1

[1] `https://www.merl.com/brdf/`



Obrázek 2.1: Ukázka aplikace Disney BRDF Explorer [1]

Článek [3] se zabývá implementací systému pro vizualizaci BRDF funkcí dostupných v databázi „Oregon BRDF Library“[2]. Je možné vizualizovat i uživatelsky zadané funkce, je potřeba dodržet interní formát definovaný knihovnou Oregon BRDF Library. Samotná vizualizace je interaktivní, je možné za běhu měnit parametry funkcí, přepínat mezi jednotlivými BRDF funkcemi a porovnávat je. Aplikace neumožňuje vizualizovat výsledný render modelu s použitím zvolené BRDF funkce. Aplikace je implementována za pomocí knihovny AVS Express. Výsledná aplikace není veřejně dostupná. Ukázka aplikace je zobrazena na obrázku 2.2.

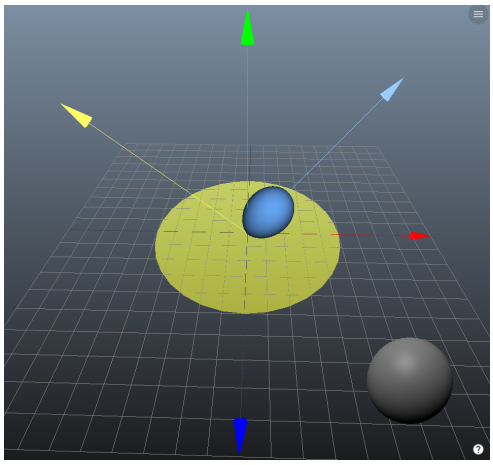


Obrázek 2.2: Ukázka aplikace BRDFvis [3]

Existují také aplikace, které je možné používat v prohlížeči. Taková aplikace je např. Photongram[3]. Aplikace umožňuje zobrazení čtyř základních BRDF funkcí. Jednotlivým funkcím lze nastavovat parametry, je také možné zobrazit finální render. Není však možné zobrazit finální ren-

[2] `https://math.nist.gov/~FHunt/appearance/obl.html`
[3] `http://shihchinw.github.io/Photongram`

der i vizualizaci BRDF funkce zároveň. Aplikace má dle mého názoru neintuitivní ovládání. Mezi jednotlivými prvky aplikace je přepínání řešeno pomocí klávesových zkratek, které při prvotním spuštění nejsou nijak představeny. Je možné se k nim dostat pouze pomocí úmyslného kliknutí na tlačítko s nápovědou. Nastavení úhlu vstupního paprsku je realizováno pomocí nastavení na jednotkové kouli. Pouhé použití tohoto způsobu nastavení je nepřesné a neintuitivní. Aplikace je v aktivním vývoji a nefunguje stabilně. Ukázka vizualizace BRDF funkce pomocí aplikace Photongram je zobrazena na obrázku 2.3.



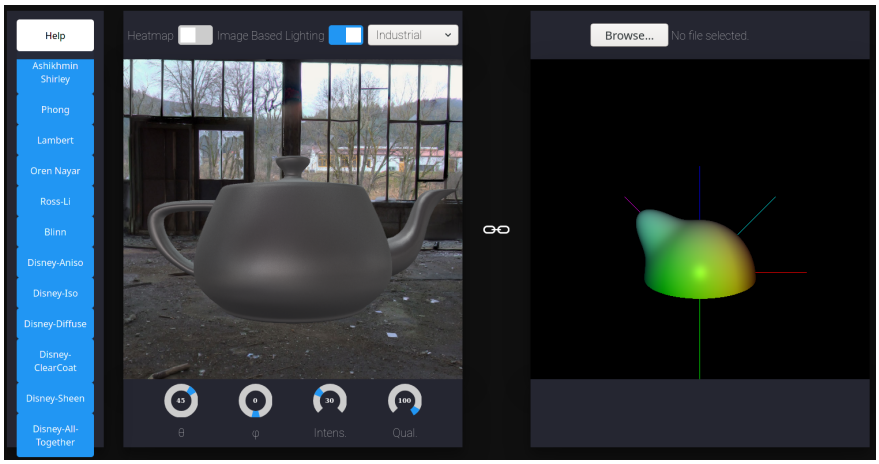
Obrázek 2.3: Ukázka aplikace Photongram

Další webovou službou pro vizualizaci BRDF je BSDF Visualizer[4]. Aplikace podporuje zobrazení základních BRDF funkcí, větší část zobrazovaných funkcí jsou různé modifikace BRDF podle Disney. Je také možné načíst uživatelsky implementovanou BRDF funkci, definovanou kódem pro GLSL shader. Načtený kód pro BRDF funkci je třeba dodat ve formátu „yaml“, je možné přidat nejen implementaci BRDF funkce spolu s pomocnými funkcemi, ale i parametry pro funkci ve formátu uniformních proměnných. Aplikace umožňuje také vizualizaci finálního renderu, který může být osvětlen pomocí bodového světla, nebo pomocí sférické mapy. Zajímavou funkcí pak je možnost přímé vizualizace BRDF funkce na povrchu renderovaného tělesa. Ukázka aplikace je zobrazena na obrázku 2.4.

2.2 Aplikace pro návrh BRDF funkcí

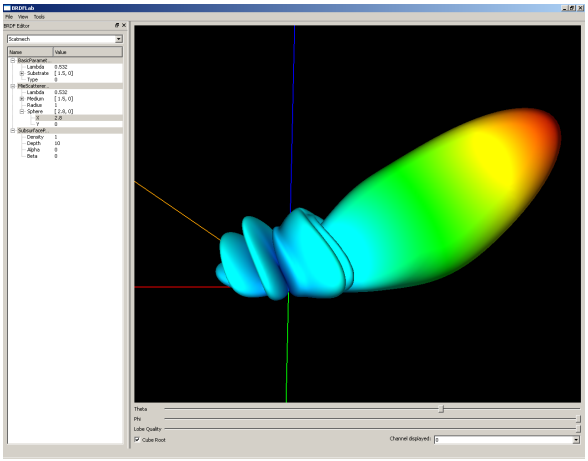
Ve článku [4] je popsána aplikace BRDFLab [5] pro vizualizaci BRDF funkcí zadaných analyticky, pomocí naměřených dat z MERL databáze, nebo pomocí dat získaných simulací. Také je možné vytvořit BRDF funkce kombinací analyticky zadaných, předpřipravených laloků (např. Lambert, Phong…). Aplikace také umožňuje vizualizaci finálního renderu s použitím BRDF funkce. Není však možné zobrazit zároveň vizualizaci BRDF funkce a finální render. Zajímavou funkcionalitou je

[4] `https://n8vm.github.io/BSDF-Visualizer/`



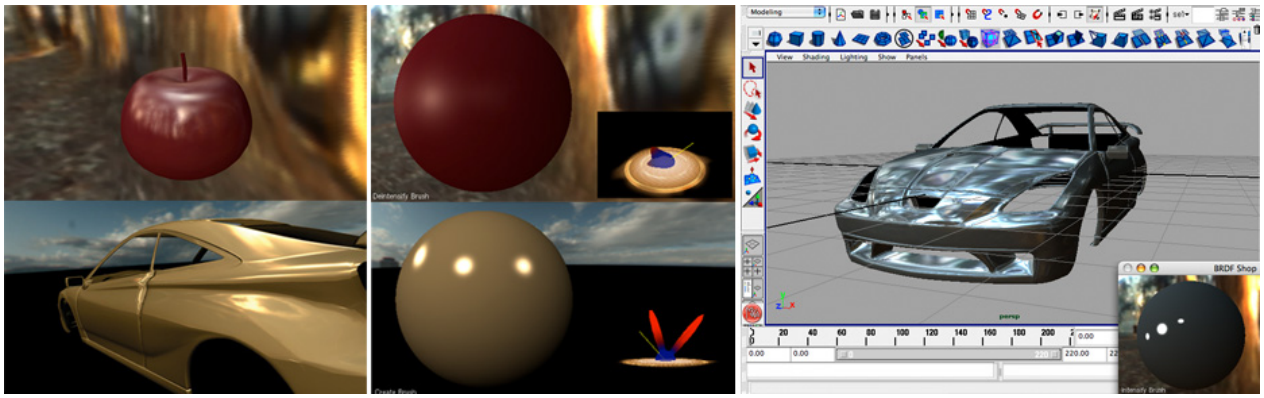
Obrázek 2.4: Ukázka aplikace BSDF Visualizer

tzv. „fitting“ – možnost aproximace komplexních BRDF funkcí (např. z MERL databáze) pomocí kombinace analyticky zadaných modelů. Při načtení naměřených dat Aplikace je distribuována jako open-source, pro GUI je použita knihovna QT, pro rendering je použita knihovna Ogre3D. Ukázka aplikace je zobrazena na obrázku 2.5.



Obrázek 2.5: Ukázka aplikace BRDFLab [4]

Článek [6] se zabývá popisem aplikace BRDF-Shop, která je primárně určena pro návrh BRDF funkcí založených na Wardově BRDF. Aplikace jako taková je implementována jako plugin do 3D editoru Autodesk Maya. Aplikace nemá možnost zadání BRDF funkce analyticky, lze pouze vytvářet nové BRDF funkce s odlesky založenými na laloku Wardovy BRDF funkce. Aplikace je primárně určena grafikům pro tvorbu materiálů, není primárně určena pro vizualizaci. Vytvořená BRDF funkce je však standardně vizualizována. Ukázka aplikace je zobrazena na obrázku 2.6.



Obrázek 2.6: Ukázka aplikace BRDF-Shop

2.3 Shrnutí

Jednotlivé aplikace pro vizualizaci a návrh BRDF funkcí mají z velké části společnou funkcionalitu a liší se hlavně v detailech. Aplikace se zásadně liší podle způsobu použití na aplikace sloužící čistě k vizualizaci BRDF funkcí a aplikace sloužící primárně pro návrh BRDF funkcí. Všechny aplikace v zásadě přistupují k samotné vizualizaci podobně, výsledky vizualizací jsou srovnatelné. Některé aplikace umožňují zobrazení BRDF funkce s interaktivní změnou parametrů zároveň i se zobrazením výsledného renderovaného obrázku. Některé aplikace také umožňují načíst uživatelsky zadané BRDF funkce. Žádná ze zde implementovaných aplikací nenabízí možnost zobrazení vzorkovacích funkcí, které jsou při použití BRDF funkcí důležité.

2.4 Požadavky na implementovanou aplikaci

Implementovaná aplikace by měla mít možnost zobrazení vybraných analyticky zadaných BRDF funkcí. Zobrazení by mělo být interaktivní, měla by být možná změna parametrů jednotlivých funkcí za běhu aplikace. Také by měla být možnost zobrazení renderovaného modelu s použitím aktuálně zadané BRDF funkce s aktuálními parametry. Vykreslování objektu by mělo korespondovat se změnou parametrů, při změně parametrů dojde k překreslení. Objekt by měl být osvětlen pomocí HDR mapy, měla by být dostupná i možnost tzv. „furnace testu“ pro ověření fyzikální korektnosti. Vykreslování modelu by mělo být rychlé, s použitím akceleračních technik. Součástí aplikace by také měla být vizualizace vzorkovacích funkcí.

Kapitola 3

Fotorealistický rendering

V počítačové grafice existuje odjakživa snaha přiblížit se výsledky reálnému světu. Fotorealistická grafika je taková, která se tváří jako nerozeznatelná např. od fotografie. Pro fotorealismus je nutné, aby rendering co nejpřesněji simuloval fyzikální principy šíření světla. Renderingem se rozumí proces výpočtu geometrie, pozice objektů a osvětlení. V roce 1986 byl realistický rendering popsán integrální rovnicí popisující přenos světelné energie ve scéně. Zobrazovací rovnice byla popsána simultánně ve článcích [7] a [8]. Kajiya v [7] popisuje různé možnosti řešení integrálu zobrazovací rovnice, mimo jiné i pomocí Monte Carlo metody, kterou pojmenoval jako path tracing [9]. Immel, Cohen a Greenberg v článku [8] navrhují řešení zobrazovací rovnice pomocí metody konečných prvků, kterou pojmenovali jako metodu radiozity. V této práci se budu dále zabývat řešením zobrazovací rovnice pouze pomocí metody Monte Carlo.

Zobrazovací rovnici lze vyjádřit vzorcem 3.1. Zobrazovací rovnice popisuje globální šíření světla ve scéně v ustáleném stavu

$$L_o\left(p,\omega_o\right) = L_e\left(p,\omega_i\right) + \int_{H(p)} L_o\left(r\left(p,\omega_o\right),-\omega_i\right) f_r\left(p,\omega_o,\omega_i\right)\left(\omega_i\cdot\hat{n}\right)\, d\omega_i\,. \tag{3.1}$$

Pro popis odrazu světla slouží rovnice odrazu, definovaná vzorcem 3.2. Tuto rovnici je možné použít pro výpočet odražené záře od objektu, pokud je známá zář dopadající na objekt

$$L_r\left(p,\omega_o\right) = \int_{H(p)} L_i\left(p,\omega_i\right) f_r\left(p,\omega_o,\omega_i\right)\left(\omega_i\cdot\hat{n}\right)\, d\omega_i\,. \tag{3.2}$$

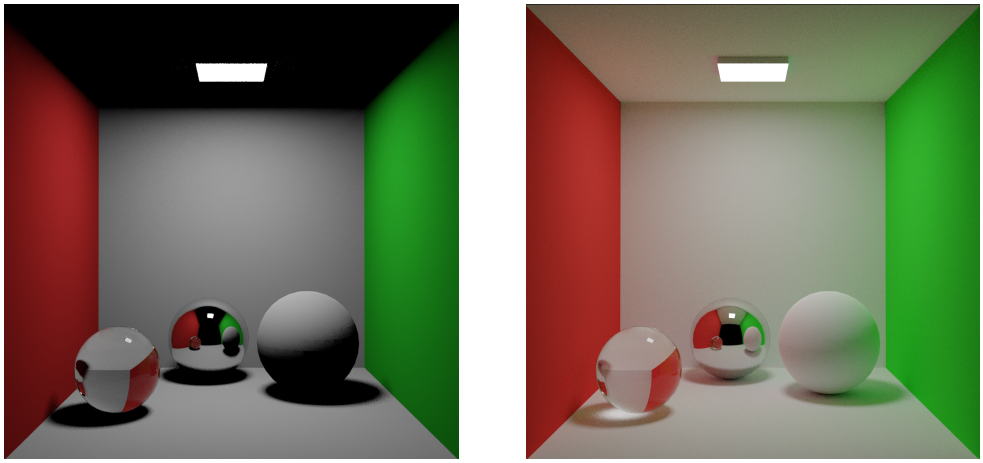
Rovnice odrazu bez BRDF členu a bez emisivního členu vyjadřuje ozáření daného bodu (viz vzorec 3.3) [10]

$$E\left(p\right) = \int_{H(p)} L_i\left(p,\omega_i\right)\left(\omega_i\cdot\hat{n}\right)\, d\omega_i\,. \tag{3.3}$$

3.1 Osvětlení

Jak už bylo zmíněno, realistický rendering se zabývá co nejpřesnější simulací šíření světla ve scéně. V reálném světě je světlo jako takové elektromagnetické záření. Viditelná část světla odpovídá zhruba intervalu $(390, 760)\,nm$ vlnové délky. Při renderingu se často pracuje pouze s geometrickou optikou. Geometrická optika zjednodušuje při šíření světla světelné paprsky na analyticky zadané orientované úsečky, zanedbává vlastnosti vycházející z vlnové podstaty světla. Z pohledu počítačové simulace jsou světelné paprsky vyzařovány světelnými zdroji do scény. Objekty ve scéně se světelnými paprsky interagují (např. pohlcují, odráží, lámou…). Tato interakce objektu a světla definuje, jak výsledný objekt vizuálně působí.

Osvětlení objektů ve scéně je popsáno osvětlovacím modelem. Pokud daný osvětlovací model bere v potaz vliv na osvětlení pouze přímými zdroji světla, jedná se o lokální osvětlovací model. Pokud osvětlovací model bere v potaz i světlo odražené ostatními objekty ve scéně, které samy o sobě žádné světlo nevyzařují, jedná se o globální osvětlovací model. Rozdíl mezi osvětlením lokálním a globálním osvětlovacím modelem je patrný z obrázku 3.1[1].



(a) Lokální osvětlovací model (b) Globální osvětlovací model

Obrázek 3.1: Srovnání osvětlovacích modelů

3.1.1 Osvětlení pomocí mapy prostředí

V počítačové grafice je v zásadě standard použití mapy prostředí jako pozadí pro scénu. Toto použití je pak dále rozšířeno v realistickém renderingu, mapa prostředí pak slouží pro osvětlení. Mapa prostředí pak v zásadě určuje osvětlení jednoho bodu ve scéně okolním prostředím, které je uloženo jako obrazová data. Reprezentace osvětlení z pohledu jednoho bodu je zobrazeno na obrázku 3.2a. V praktické implementaci se pak většinou používá sférická mapa (lze se také setkat

[1] Výsledné renderované obrázky byly vytvořeny v programu Blender 2.92

s pojmem tzv. „latitude, longitude“ mapování), která je aplikována jako textura na pomyslnou kouli obklopující scénu, nebo tzv. „cube map“, kde je mapa použita jako textura pro pomyslnou krychli obklopující scénu. Ukázka sférické mapy je zobrazena na obrázku 3.2b, ukázka cube map je ukázána na obrázku 3.2c. Sférická mapa je častěji používána s technologií ray tracing, z důvodu jednoduššího převodu směrového vektoru sledovaného paprsku do sférických souřadnic a následně do samotného texelu této textury. Cube map je pak častěji používána v klasické rasterizaci z důvodu jednoduchého použití díky přímé podpoře v OpenGL shaderech.

Fotorealistického osvětlení modelů ve scéně pomocí mapy prostředí je dosaženo díky odrazům paprsků od povrchu objektů, které jsou odraženy do mapy prostředí. Texely mapy prostředí tak mají přímý vliv na osvětlení objektu a není potřeba použít ve scéně bodová světla. Pro fotorealismus je také důležitý formát mapy prostředí. Standardní fotografie nemají dostatečnou bitovou hloubku pro zachycení reálné intenzity světla. Je proto vhodné použít HDR snímky, kde jsou hodnoty jednotlivých pixelů reprezentovány jako reálná čísla ve formátu float.

(a) (b) (c)

Obrázek 3.2: Ukázka různých map prostředí [11]

3.2 Rendering pomocí metody sledování paprsku

Při renderingu pomocí klasických metod se postupuje pomocí standardního zobrazovacího řetězce. Postup je ve zkratce následující: Vrcholy tělesa jsou nahrány do GPU, následně se ve fragment shaderu aplikují transformační matice a matice pro převod z lokálního do globálního prostoru. Následně je provedena rasterizace tělesa. Nakonec je ve fragment shaderu vypočítáno osvětlení tělesa. Standardní zobrazovací řetězec má tu výhodu, že je možné renderovat za běhu programu, v reálném čase. Nevýhodou je však ztráta jisté reálnosti výstupu.

Při použití metody sledování paprsku je perspektivní promítání realizováno pomocí sady přímek (paprsků). Paprsky jsou konstruovány takovým způsobem, že procházejí středem promítání. Směry těchto přímek pak korespondují s rozmístěním jednotlivých pixelů generovaného obrazu na pomyslné promítací rovině. Následně se provádí traverzace ve scéně (hledání objektu ve scéně, který byl protnut paprskem). Při protnutí tělesa je možné následně počítat další odrazy paprsků a simulovat tak šíření světla ve scéně. Výhodou použití metody sledování paprsků je možnost velmi reálných

výsledků. Nevýhodou je ale vysoká výpočetní náročnost. Výpočet komplexních scén pouze pomocí metod sledování paprsků v reálném čase je komplikovaný problém. V současné době je možná hardwarová akcelerace na grafických kartách. Pro urychlení výpočtu jsou také používány hluboké neuronové sítě (např. technologie Nvidia DLSS). Je možné provádět výpočet pro nižší rozlišení a následně provést upscaling (zvýšení rozlišení). Neuronová síť je schopná dopočítat chybějící data. Výsledkem je dle Nvidia obraz svou kvalitou srovnatelný s výpočtem ve vyšším rozlišení bez použití této technologie.

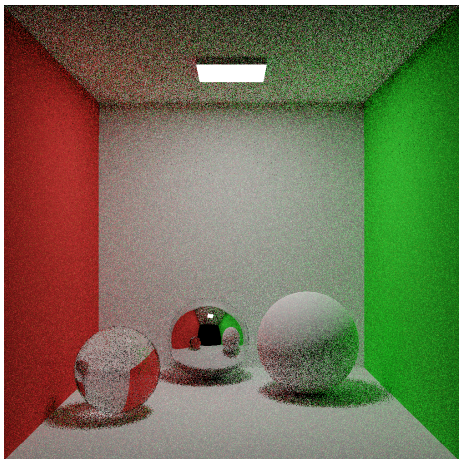
3.3 Řešení zobrazovací rovnice

Zobrazovací rovnice má formu integrálu, který se počítá přes pomyslnou polokouli vytvořenou nad bodem, orientovanou podél normály bodu. Na výslednou hodnotu má vliv nekonečné množství příchozích paprsků ze všech směrů na povrchu polokoule. Zář $L_i(p, \omega_i)$ dopadající na bod mohlo být odraženo jiným bodem ve scéně. Z toho vyplývá, že výpočet osvětlení je nekonečně rekurzivní. Korektní analytické řešení zobrazovací rovnice z toho důvodu není možné. Z matematického hlediska existuje více způsobů, jak řešit integrály. Metoda Monte Carlo se opírá o princip z teorie pravděpodobnosti a matematické statistiky, že průměr velkého počtu náhodných veličin se přiblíží střední hodnotě. Řešení zobrazovací rovnice pomocí metody Monte Carlo navrhuje Kajiya v [7].

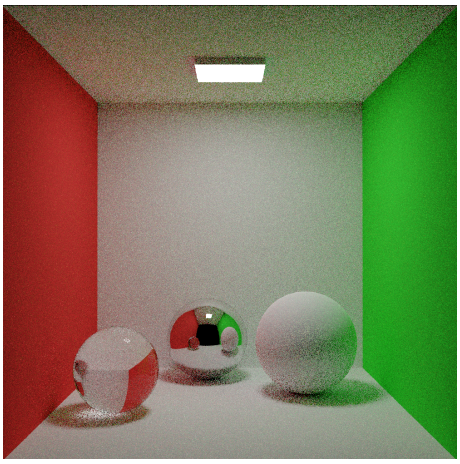
Monte Carlo je stochastická metoda, v matematice často využívaná pro řešení složitějších integrálů. Metoda Monte Carlo řeší integrály modelováním náhodných veličin a následným odhadem jejich charakteristik. Nejčastěji tato metoda modeluje náhodné veličiny tak, že se střední hodnota rovná hledané hodnotě [12].

Pro řešení zobrazovací rovnice jsou typicky generovány náhodné směry odrazu světla. Výsledný obraz je poté tvořen průměrem určitého počtu vzorků. Typicky se u výsledných obrázků vytvořených pomocí metody path tracing uvádí počet vzorků na pixel. Se zvyšujícím se počtem vzorků na pixel typicky stoupá ostrost a přesnost výsledného obrazu. S nízkým počtem vzorků je typicky obraz zatížen šumem (viz srovnání na obrázku 3.3)[2].

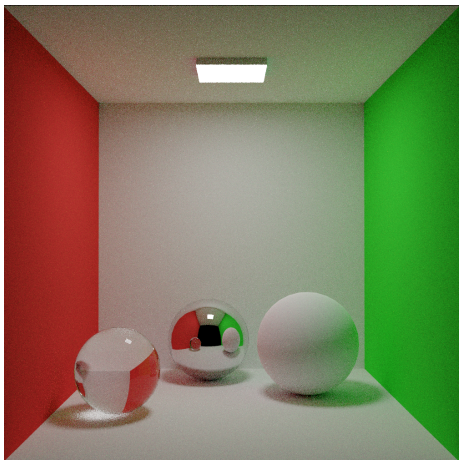
[2]Výsledné renderované obrázky byly vytvořeny v programu Blender 2.92



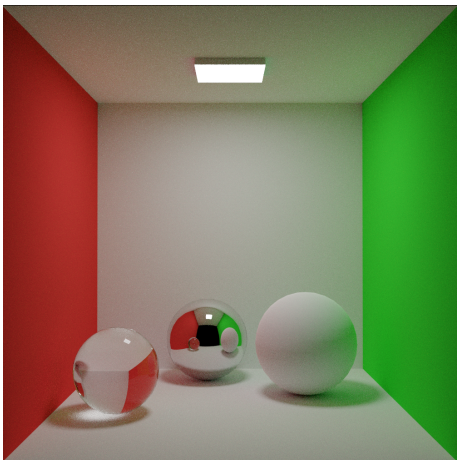
(a) 1 vzorek na pixel



(b) 4 vzorky na pixel



(c) 16 vzorků na pixel



(d) 64 vzorků na pixel

Obrázek 3.3: Srovnání výsledných obrázků podle počtu vzorků na pixel. S vyšším počtem vzorků na pixel se dramaticky snižuje šum.

Kapitola 4

BRDF

BRDF je matematická funkce, která definuje pro daný materiál odrazivost povrchu. Určuje pro každý bod tělesa distribuci odrazu světla. Na obrázku 4.1 je znázorněn základní princip BRDF funkce. Vektor $\hat{n}$ je normála povrchu, ω_i značí směr ke světelnému zdroji, ω_o značí směr k pozorovateli (ke kameře). BRDF je definována pro dvojici vektorů ω_i a ω_o v daném bodě p pomocí vzorce 4.1

$$f_r\left(p, \omega_o, \omega_i\right) = \frac{dL_o\left(p, \omega_o\right)}{dE\left(p, \omega_i\right)} = \frac{dL_o\left(p, \omega_o\right)}{L_i\left(p, \omega_i\right)\left(\omega_i \cdot \hat{n}\right) d\omega_i}\,. \tag{4.1}$$

Je žádoucí, aby BRDF funkce splňovaly některá základní pravidla. Je důležitá fyzikální přesnost BRDF funkce, pro kterou musí být splněny následující podmínky [13]:

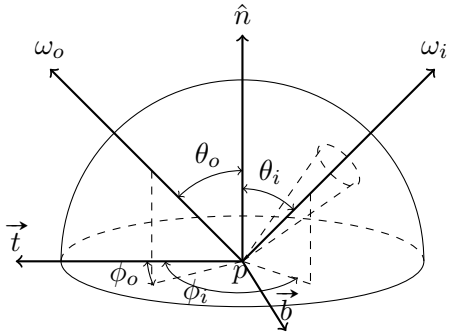
- Princip vzájemnosti (Helmholtzův princip reciprocity, [14]) – Pro všechny dvojice ω_i a ω_o platí: $f_r\left(p, \omega_o, \omega_i\right) = f_r\left(p, \omega_i, \omega_o\right)$.
- Princip zachování energie – Celková energie odraženého světla nemůže být vyšší než energie příchozího světla. Zároveň je také žádoucí, aby bylo odraženo a pohlceno všechno příchozí světlo.
- BRDF musí mít vždy nezáporný výsledek.

Tyto podmínky jsou potřeba pro korektní výstup BRDF funkce a je možné je považovat za základní trojici podmínek. Dále je možné definovat několik dalších podmínek pro BRDF funkce, aby výsledné rendery byly věrohodné z vizuálního hlediska [15]:

- BRDF funkce by měla být spojitá. Nespojitosti jsou pro lidské oko nepřirozené. Spojitost BRDF funkcí mohou porušovat např. BRDF funkce, které využívají metody „min“, „max“.
- BRDF funkce by měla mít možnost nastavení parametru zrcadlové složky tak, aby bylo možné dosáhnout i matného povrchu.

- BRDF funkce by měla mít možnost nastavení parametru zrcadlové složky tak, že pokud je hodnota tohoto parametru 1, BRDF funkce se blíží ideálnímu odrazu.
- Možnost jednoduchého vzorkování – BRDF funkce by měla být jednoduše vzorkovatelná, pdf funkce pro BRDF funkci by měla být analyticky vyjádřitelná.

Další doplňující podmínky lze definovat pro BRDF funkce, které využívají mikroplošky. Podmínky pro BRDF využívající mikroplošky jsou podrobněji rozvedeny v kapitole 4.2.4 Torrance-Sparrow.



Obrázek 4.1: Základní vizualizace BRDF funkce

4.1 Variace BRDF funkcí

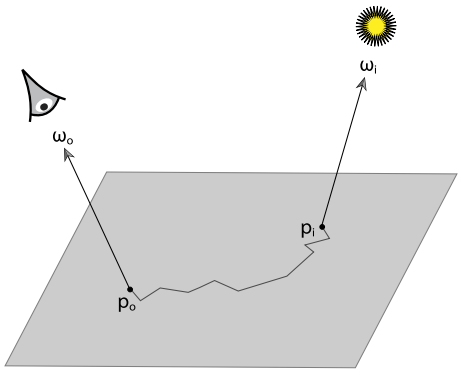
Kromě BRDF funkcí jsou v počítačové grafice definovány i další funkce, které se používají pro popis odrazu světla. BTDF funkce (Bidirectional Transmittance Distribution Function) je distribuční funkce lomu světla, která popisuje průchod světla objektem. BTDF funkce je v zásadě funkce $f_t(p, \omega_o, \omega_i)$, pro kterou platí, že vektory ω_i a ω_o směřují každý do opačné hemisféry bodu p.

BSDF funkce (Bidirectional scattering distribution function) vzniká v zásadě spojením BRDF a BDTF funkce a popisuje rozptyl světla při kontaktu s objektem.

Pokročilejší funkcí je tzv. BSSRDF funkce (Bidirectional sub-surface scattering reflectance distribution function). BRDF funkce odráží příchozí světlo vždy ve stejném bodě, BSSRDF funkce modeluje průchod světla a odrazy pod povrchem tělesa. Princip a výpočet BSSRDF funkcí je netriviální, ale použití výsledné funkce má pozitivní dopad na výslednou realističnost scény. Ukázka odrazu světla pomocí BSSRDF funkce je zobrazena na obrázku 4.2.

4.2 Přehled BRDF funkcí

Následující odstavce se zabývají detailním popisem a vlastnostmi jednotlivých BRDF funkcí. Budou zde vyjmenovány pouze ty BRDF funkce, které byly implementovány v praktické části této



Obrázek 4.2: Ukázka principu BSSRDF funkce [13]

diplomové práce.
Pro rychlý přehled vlastností jednotlivých BRDF funkcí je možné využít tabulku 4.1 (převzato z [16], zkráceno, doplněno).

BRDF model	Physical	Plausible	Fresnel Eq.	Anisotropic	Sampling	Rel. cost (cycles)	Material type
Ideal reflection	✓	✓	✗	✗	✓	1	Mirror, perfect spec.
Lambert	✓	✓	✗	✗	✓	1	Perfect diffuse
Phong	✗	✗	✗	✗	✓	...	Rough surf.
Blinn-Phong	✗	✗	✗	✗	✓	9,18	Rough surf.
Phys. correct Phong	✗	✓	✗	✗	✓	...	Rough surf.
Torrance-Sparrow	✓	✗	✓	✓	✗	...	Rough surf.
Cook-Torrance	✓	✓	✓	✗	✗	16,90	Metal, plastic
Oren-Nayar	✓	✓	✗	✗	✓	10,98	Matte, dirty

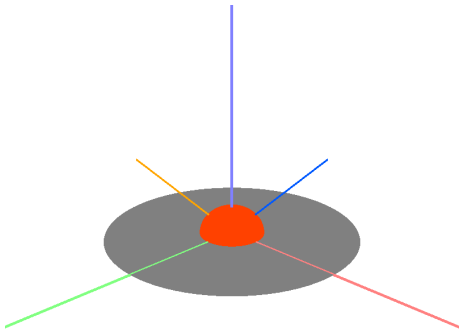
Tabulka 4.1: Stručný přehled implementovaných BRDF funkcí [16]

4.2.1 Lambert

BRDF podle Beer-Lambertova zákona se řadí mezi analytické modely BRDF. Popisuje ideálně matné povrchy, které odráží příchozí světlo do všech směrů rovnoměrně se stejnou pravděpodobností, nehledě na příchozí směr paprsku. Jedná se o nejjednodušší BRDF funkci, je definována vztahem 4.2 [17]

$$f_r\left(p, \omega_o, \omega_i\right) = \frac{\rho}{\pi} = \text{konstantní}\,. \tag{4.2}$$

Parametr ρ ve vztahu 4.2 vyjadřuje poměr mezi pohlceným a odraženým světlem. Tato veličina je také odborně nazývána pojmem albedo. Dělení konstantou π zajišťuje platnost zákona zachování energie. Díky nezávislosti na směru vstupního paprsku je také splněna reciprocita. Ukázka vizualizace je zobrazena na obrázku 4.3 s parametrem $\rho = 0{,}8$.



Obrázek 4.3: Vizualizace BRDF podle Beer-Lambertova zákona

4.2.2 Ideální odraz

BRDF pro ideální odraz odráží světlo pouze v jednom směru podle zákona odrazu světla. BRDF funkce pro ideální odraz je definována jako Diracova δ-funkce, viz vzorec 4.3. Výsledkem je poté zrcadlový materiál. Ukázka vizualizace je zobrazena na obrázku 4.4.

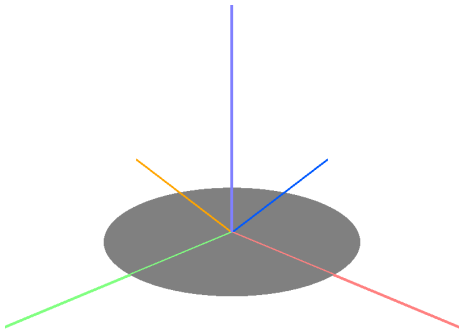
$$f_r\left(p, \omega_o, \omega_i\right) = \begin{cases} \infty & \theta_i = \theta_o \\ 0 & \theta_i \neq \theta_o \end{cases} = \frac{\delta\left(\cos\theta_i - \cos\theta_o\right)\delta\left(\phi_i - \phi_o + \pi\right)}{\cos\theta_i} \tag{4.3}$$

4.2.3 Phong

BRDF podle Phongova modelu vychází z Phongova osvětlovacího modelu, řadí se mezi analytické modely BRDF a používá se pro lesklé povrchy. Původní model je definován vzorcem 4.4 [18]. Ukázka vizualizace je zobrazena na obrázku 4.5a s parametrem $n = 32$

$$f_r\left(p, \omega_o, \omega_i\right) = k_s(\omega_r \cdot \omega_o)^n\,. \tag{4.4}$$

Upravený, také často využívaný Blinn-Phong model, je definován vzorcem 4.5 [19]. Výhodou upraveného Blinn-Phongova modelu oproti klasickému Phongovu modelu je hladší přechod odlesku



Obrázek 4.4: Vizualizace BRDF ideálního odrazu

světla. Ukázka vizualizace je zobrazena na obrázku 4.5b s parametrem $n = 32$

$$f_r\left(p, \omega_o, \omega_i\right) = k_s(\hat{n} \cdot \hat{h})^n . \tag{4.5}$$

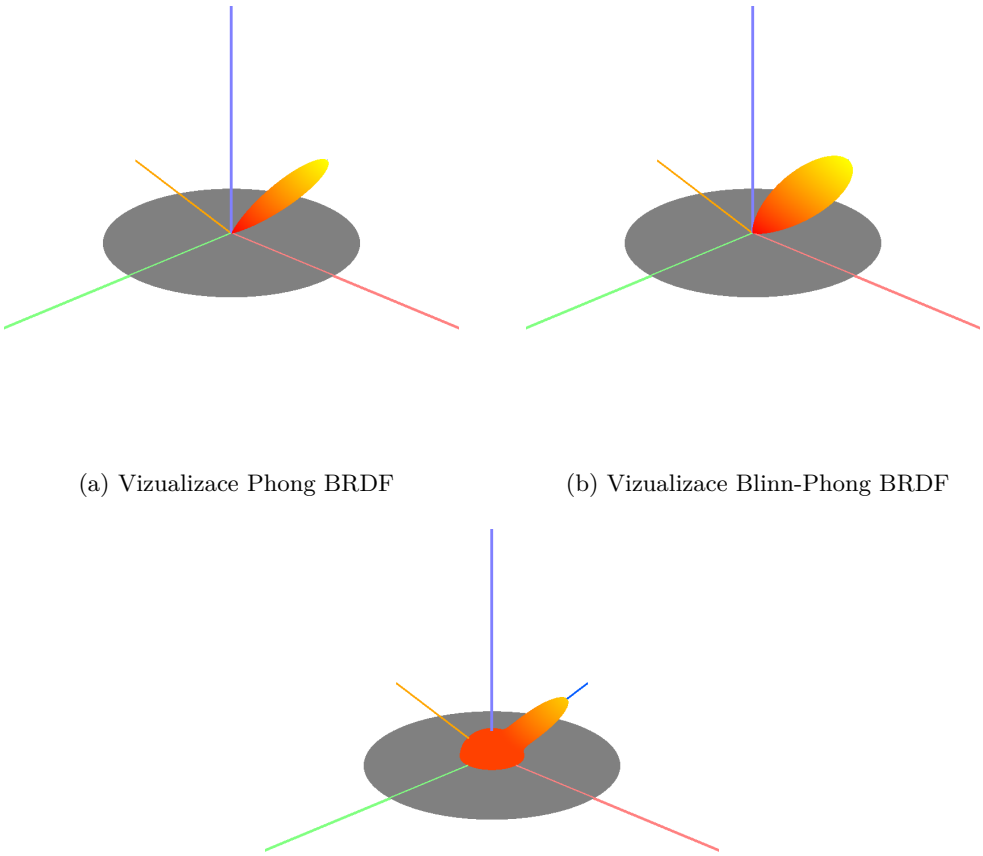
Jak Phong, tak Blinn-Phong modely nejsou fyzikálně přesné – nesplňují zákon zachování energie, ani zákon reciprocity [16]. BRDF podle Phongova modelu je možné dále upravit, aby byly splněny pravidla pro fyzikální korektnost. Fyzikálně korektní BRDF podle Phongova modelu je definováno vzorcem 4.6. Pro splnění podmínky zachování energie této varianty je potřeba dodržet $k_d + k_s \leq 1$ [20]. Tato varianta je stále empirická a fyzikálně korektní je pouze při kolmém pohledu na plošku, pro ostatní směry je výsledná energie nižší. Ukázka vizualizace je zobrazena na obrázku 4.5c s parametry $n = 32,\ k_s = 0{,}1,\ k_d = 0{,}8$

$$f_r\left(p, \omega_o, \omega_i\right) = \frac{k_d}{\pi} + \frac{k_s\left(n+2\right)}{2\pi}(\cos\theta_r)^n . \tag{4.6}$$

4.2.4 Torrance-Sparrow

Torrance-Sparrow BRDF patří mezi fyzikální modely a je považován za jeden s nejúplnějších modelů [16]. Mimo jiné je např. schopen simulovat odlesk polarizovaného světla. Tento model simuluje mikroploškové materiály a pomocí parametru roughness (drsnost) simuluje mikroskopické nerovnosti materiálu. Orientace mikroskopických nerovností je v materiálu náhodná. Vyšší hodnota drsnosti materiálu snižuje lesklost materiálu. Torrance-Sparrow BRDF funkce je definována vzorcem 4.7. Ukázka vizualizace je zobrazena na obrázku 4.6 s parametry $\sigma = 0{,}35,\ F_0 = 1$

$$f_r\left(p, \omega_o, \omega_i\right) = \frac{k_d}{\pi} + \frac{k_s D(\hat{h}, \sigma) F(\omega_o) G(\omega_o, \omega_i, \hat{m})}{4\pi(\hat{n} \cdot \omega_i)} . \tag{4.7}$$



(a) Vizualizace Phong BRDF

(b) Vizualizace Blinn-Phong BRDF

(c) Vizualizace fyzikálně korektní verze BRDF podle Phongova modelu

Obrázek 4.5: Vizualizace jednotlivých verzí BRDF podle Phongova modelu

Pro výpočet se používá distribuční funkce D, která popisuje rozložení normál mikroplošek. V této konkrétní implementaci je použita Beckmannova distribuční funkce

$$D(\hat{h}, \sigma) = \frac{e^{\left(\frac{(\hat{n}\cdot\hat{h})^2-1}{\sigma^2(\hat{n}\cdot\hat{h})^2}\right)}}{\sigma^2(\hat{n}\cdot\hat{h})^2}\,. \tag{4.8}$$

Beckmannova distribuční funkce pracuje s normálou mikroplošky (kolem které se generuje rozložení) a hodnotou drsnosti materiálu. Jako normála mikroplošky je použit poloviční vektor mezi pohledovým a světelným vektorem (viz 4.2.3 Phong) z důvodu, že mikroploška perfektně odráží světlo právě v případě, kdy je normála mikroplošky shodná s polovičním vektorem $\hat{h}$ [21].

Dále se počítá poměr odraženého světla a lomeného světla F pomocí Fresnelových vzorců. Pro výpočet je použita Schlickova aproximace 4.9 [22]

$$F(\omega_o) \approx R(\theta) = F_0 + (1 - F_0)(1 - \cos\theta)^5 . \tag{4.9}$$

Poslední část pro výpočet je koeficient geometrického útlumu G, který vyjadřuje zakrytí mikroplošek při odrazu světla 4.10 [16]

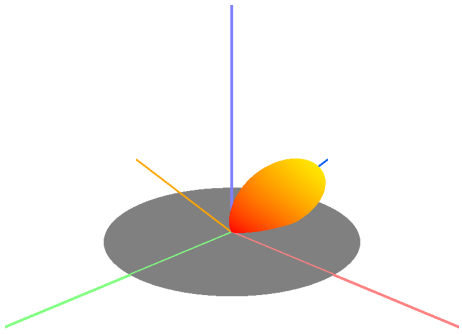
$$G(\omega_o, \omega_i, \hat{m}) = \min\left(1, \frac{2(\hat{n}\cdot\hat{h})(\hat{n}\cdot\omega_o)}{(\omega_o\cdot\hat{h})}, \frac{2(\hat{n}\cdot\hat{h})(\hat{n}\cdot\omega_i)}{(\omega_o\cdot\hat{h})}\right) . \tag{4.10}$$

Ve článku [23] byly uvedeny podmínky pro distribuční funkce D a funkce pro výpočet geometrického útlumu G, které by měly být splněny, aby byly výsledky realistické. Pro distribuční funkce D by mělo být splněno:

- Hodnota distribuční funkce je nezáporná, $D(\hat{h}, \sigma) \in \langle 0, \infty)$.
- Celková plocha mikroplošek povrchu tělesa je větší nebo rovna ploše povrchu tělesa.
- Zobrazená plocha mikroplošek je stejná jako zobrazená plocha povrchu tělesa.

Funkce geometrického útlumu G by měly splňovat alespoň následující podmínky:

- Pro hodnotu funkce geometrického útlumu platí: $G(\omega_o, \omega_i, \hat{m}) \in \langle 0, 1\rangle$.
- Funkce útlumu by měla být symetrická, platí $G(\omega_o, \omega_i, \hat{m}) = G(\omega_i, \omega_o, \hat{m})$.
- Zadní strana mikroplošky by neměla být viditelná v případě pohledu na přední stranu makroplošky a naopak.



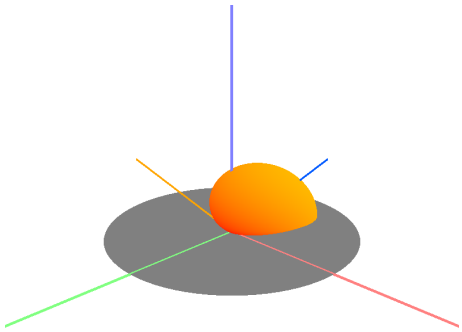
Obrázek 4.6: Vizualizace Torrance-Sparrow BRDF

4.2.5 Cook-Torrance

Cook-Torrance BRDF rozšiřuje mikroplošková BRDF o myšlenku, že pouze mikroplošky orientované podél vektoru $\hat{h}$ mají vliv na výsledný odraz světla. Pro výpočet odrazové složky jsou využity opět funkce F, D, G (viz 4.2.4 Torrance-Sparrow). Cook-Torrance BRDF se vypočítá vzorcem 4.11 [24]

$$f_r\left(p, \omega_o, \omega_i\right) = \frac{F(\omega_o)D(\hat{h}, \sigma)G(\omega_o, \omega_i)}{4(\hat{n} \cdot \omega_i)(\hat{n} \cdot \omega_o)}\,. \tag{4.11}$$

Nevýhodou této BRDF funkce je nesplnění fyzikální přesnosti z důvodu nesplnění zákona zachování energie pro některé (ω_o, ω_i) [16]. Ukázka vizualizace je zobrazena na obrázku 4.7 s parametry $\sigma = 0{,}7,\ F_0 = 0{,}6$.



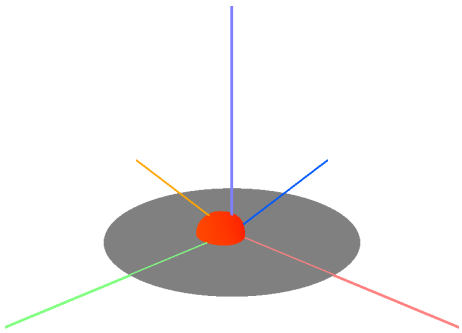
Obrázek 4.7: Vizualizace Cook-Torrance BRDF

4.2.6 Oren-Nayar

Oren-Nayar BRDF popisuje Lambertovské difusivní materiály. Na rozdíl od BRDF podle Beer-Lambertova zákona (viz 4.2.1 Lambert) pracuje Oren-Nayar BRDF s mikroploškami, které si lze představit jako mikroskopické nerovnosti ve tvaru písmena „V“. Dále, na rozdíl od jiných mikroplošk-ových modelů (Torrance-Sparrow) nejsou mikroplošky odrazivé, ale difuzivní. Oren-Nayar BRDF bere v úvahu odrazy světla mezi jednotlivými mikroploškami s limitovaným maximálním počtem odrazů mezi dvojicí mikroplošek. Pro generování distribuce orientace mikroplošek je použito Gaussovo rozdělení, které je nastavena pomocí parametru σ. Parametr σ určuje směrodatnou odchylku úhlu orientace mikroplošky. [16, 25]. Oren-Nayar BRDF je definováno vzorcem 4.12

$$
\begin{aligned}
\alpha &= \max(\theta_i, \theta_o) = \max(\arccos(\hat{n} \cdot \omega_i), \arccos(\hat{n} \cdot \omega_o)) \\
\beta &= \min(\theta_i, \theta_o) = \min(\arccos(\hat{n} \cdot \omega_i), \arccos(\hat{n} \cdot \omega_o)) \\
\cos(\phi_r - \phi_i) &= (\omega_i - \widehat{\hat{n}(\hat{n} \cdot \omega_i)}) \cdot (\omega_o - \widehat{\hat{n}(\hat{n} \cdot \omega_o)}) \\
C_1 &= 1 - 0{,}5\frac{\sigma^2}{\sigma^2 + 0{,}33} \\
C_2 &= \begin{cases} 0{,}45\frac{\sigma^2}{\sigma^2+0{,}09}\sin\alpha & \cos(\phi_r - \phi_i) \geq 0 \\ 0{,}45\frac{\sigma^2}{\sigma^2+0{,}09}\left(\sin\alpha - \left(\frac{2\beta}{\pi}\right)^3\right) & jinak \end{cases} \\
C_3 &= 0{,}125\left(\frac{\sigma^2}{\sigma^2 + 0{,}009}\right)\left(\frac{4\alpha\beta}{\pi^2}\right)^2 \\
L_r^1 &= \frac{\rho}{\pi}\left[C_1 + \cos(\phi_r - \phi_i)\, C_2 \tan\beta + (1 - |\cos(\phi_r - \phi_i)|)\, C_3 \tan\left(\frac{\alpha+\beta}{2}\right)\right] \\
L_r^2 &= 0{,}17\frac{\rho^2}{\pi}\frac{\sigma^2}{\sigma^2 + 0{,}13}\left[1 - \cos(\phi_r - \phi_i)\left(\frac{2\beta}{\pi}\right)^2\right] \\
f_r(p, \omega_o, \omega_i) &= L_r^1 + L_r^2 .
\end{aligned}
\tag{4.12}
$$

Ukázka vizualizace je zobrazena na obrázku 4.8 s parametry $\sigma = 0{,}7,\ \rho = 0{,}8$.



Obrázek 4.8: Vizualizace Oren-Nayar BRDF

4.3 Srovnání náročnosti výpočtu BRDF funkcí

Následující kapitoly se zabývají srovnáním jednotlivých BRDF funkcí dle jejich výpočetní složitosti.

4.3.1 Metodika výpočtu výpočetní složitosti

Při výpočtu složitosti jsem se rozhodl pro praktické měření výkonu jednotlivých algoritmů, protože výsledná hodnota má pro praktickou implementaci vyšší informační hodnotu než teoretická analýza. Jako metriku jsem zvolil absolutní a relativní čas výpočtu řádově 10^7 iterací výpočtu. Výsledky měření jsem pak rozdělil do tří kategorií - celkové srovnání, srovnání BRDF funkcí pro difuzní povrchy a srovnání BRDF funkcí pro lesklé povrchy.

4.3.2 Výsledky měření, zhodnocení

Konkrétní výsledky měření jsou uvedeny v tabulkách 4.2, 4.3 a A.1. Z výsledků z tabulky 4.2 vyplývá, že BRDF podle Beer-Lambertova zákona je z hlediska výpočetního výkonu nejefektivnější BRDF funkce pro matné povrchy. Tento výsledek je očekávatelný z důvodu velmi jednoduchého vzorce. Oproti BRDF podle Beer-Lambertova zákona ale Oren-Nayar poskytuje širší možnosti nastavení díky parametru drsnosti. Oren-Nayar tak může věrněji reprezentovat drsnější difuzivní materiály.

Ze srovnání BRDF funkcí pro lesklé povrchy v tabulce 4.3 vyplývá, že mezi jednotlivými složitějšími funkcemi není rozdíl příliš markantní. BRDF funkce dokonalého odrazu je opět dle očekávání nejefektivnější. Pozornost si zaslouží rozdíl mezi tradičním BRDF podle Phongova modelu a jeho fyzikálně přesnou variantou. Je vidět, že přidáním fyzikální korektnosti se výkon zhoršil pouze o cca 16 %.

BRDF Funkce	Absolutní čas	Relativní čas
Lambert	0,05 s	1,00
Oren-Nayar	1,36 s	28,36

Tabulka 4.2: Srovnání BRDF funkcí pro matné povrchy

BRDF Funkce	Absolutní čas	Relativní čas
Mirror	0,04 s	1,00
Phong	0,41 s	10,84
Physically correct Phong	0,48 s	12,79
Blinn-Phong	0,50 s	13,25
Cook-Torrance	0,53 s	13,98
Torrance-Sparrow	0,54 s	14,18

Tabulka 4.3: Srovnání BRDF funkcí pro lesklé povrchy

Kapitola 5

Vizualizační aplikace

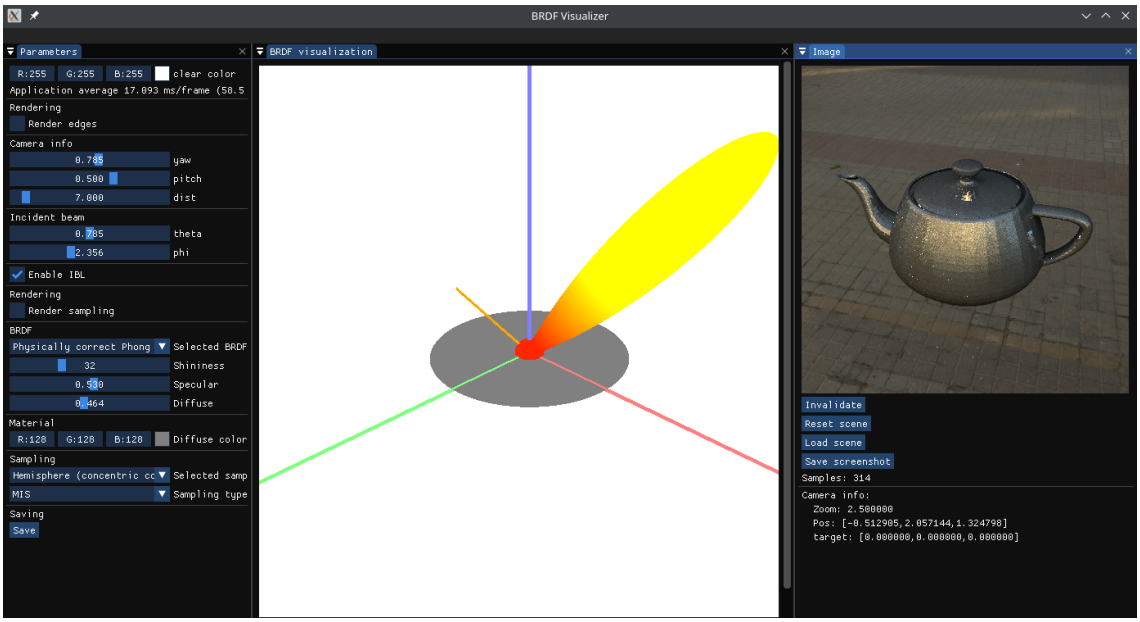
Aplikace jako taková umožňuje zobrazit BRDF funkce popsané v kapitole 4.2 Přehled BRDF funkcí. Důležitým prvkem je také interaktivita, kdy je možné měnit jednotlivé parametry BRDF funkcí. Kromě samotných BRDF funkcí aplikace umožňuje zobrazit i metody pro vzorkování funkcí (toto téma je dále rozebráno v kapitole 6 Redukce variance Monte Carlo), kdy je možné přepínat mezi jednotlivými BRDF funkcemi a vzorkovacími funkcemi. Je tedy možné demonstrovat důležitost správného výběru vzorkovací funkce k vybrané BRDF funkci. Pro doplnění funkcionality obsahuje aplikace také možnost pro uložení snímku aktuální vizualizace BRDF ve vysokém rozlišení.

Poslední funkcionalitou je interaktivní okno s jednoduchou scénou, která je renderována s použitím aktuálně vybrané BRDF funkce. Pro jednoduchost je renderován model koule, který je popsán analyticky. Tato testovací scéna je primárně osvětlena HDR obrazem, ale aplikace také umožňuje osvětlit scénu konstantním jasem. Tímto nastavením je simulována situace, kdy na objekt dopadá ze všech směrů stejné množství světla. Takto osvětlená scéna je známa jako tzv. furnace test a slouží pro kontrolu fyzikální korektnosti BRDF funkce. Ukázka uživatelského rozhraní aplikace je zobrazena na obrázku 5.1.

Následující odstavce se zabývají konkrétními detaily implementace a popisem technických řešení použitých pro jednotlivé funkce implementované vizualizační aplikace.

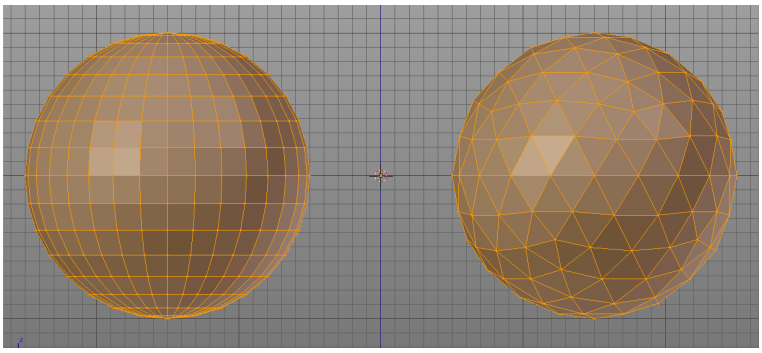
5.1 Vizualizace BRDF

Při implementaci jsem se chtěl co nejvíce přiblížit standardním referenčním obrázkům popisujícím princip BRDF funkcí (viz obrázek 4.1). Výsledná vizualizace znázorňuje pro daný vstupní směr všechny možné výstupní směry, do kterých se světlo odráží. Této vizualizace se dá jednoduše dosáhnout tak, že se vygeneruje jednotková hemisféra a jednotlivé body na hemisféře jsou posunuty o hodnotu BRDF funkce pro daný vstupní a výstupní směr. Každý bod na hemisféře zvýrazňuje výstupní směr, vzdálenost bodu od středu (délka vektoru) zvýrazňuje hodnotu BRDF funkce.



Obrázek 5.1: Uživatelské rozhraní vytvořené aplikace

Pro vizualizaci jsem se rozhodl vygenerovat jednotkovou polokouli a jednotlivé body polokoule upravit v OpenGL Vertex shaderu. Při generování polokoule jsem narazil na problém, kdy pro vizualizaci bylo potřeba mít co nejrovnoměrněji rozložené polygony, ideálně všechny s podobnou velikostí. Rozhodl jsem se tedy negenerovat UV kouli, ale geodetický mnohostěn (srovnání na obrázku 5.2), který je tvořen rovnostrannými trojúhelníky.



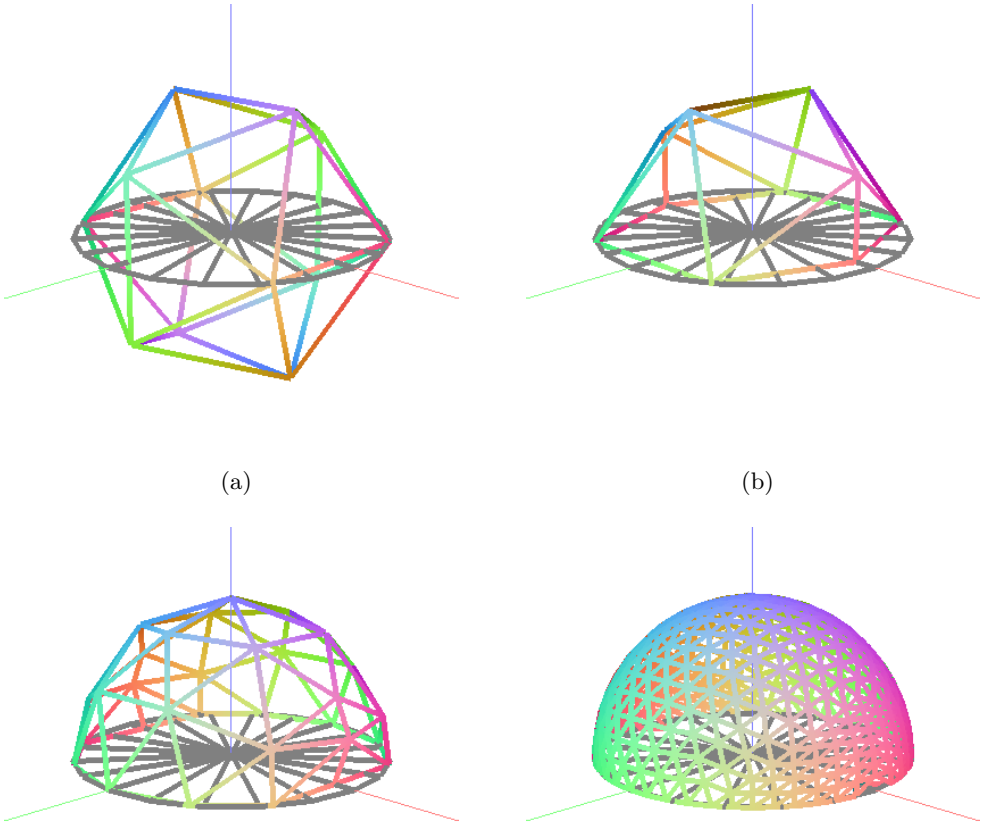
Obrázek 5.2: Srovnání UV koule a geodetického mnohostěnu [26]

Pro zjednodušení implementace jsem vycházel z kódu pro dvacetistěn [27], jehož trojúhelníkové stěny jsem dále rekurzivně dělil. Počet rekurzivních dělení určuje rozlišení výsledné vizualizace, s vyšším počtem dělení se zvyšuje rozlišení. Generování hemisféry probíhá v následujících krocích:

1. Vygenerování dvacetistěnu (viz obrázek 5.3a)
2. Rozdělení dvacetistěnu na poloviční dvacetistěn (viz obrázek 5.3b)

3. Rekurzivní dělení polygonů až do dosažení požadované jemnosti povrchu (viz obrázek 1× rozdělených polygonů 5.3c a obrázek 3× rozdělených polygonů 5.3d)

Postup generování hemisféry použité pro vykreslení BRDF funkcí je zobrazen na obrázku 5.3.



(a) (b)

(c) (d)

Obrázek 5.3: Postup generování hemisféry

Body takto generované hemisféry jsou následně upraveny ve vertex shaderu. Základní kód pro úpravu hemisféry glsl vertex shaderem je uveden ve výpise 1. Vertex shader pracuje se vstupními parametry `in_Position`, `u_incidentVector` a parametry pro jednotlivé BRDF funkce. Pro jednoduchost je výpis zkrácen a zjednodušen. Parametr `in_Position` určuje vstupní bod hemisféry a před vstupem do BRDF funkce je normalizován (aby byla pojištěna jednotková vzdálenost od středu). Parametr `u_incidentVector` představuje směrový vektor ω_o.

Ve vertex shaderu jsou poté ve funkci `BRDF` implementovány jednotlivé BRDF funkce. Pro přepínání mezi aktuálně zvolenou BRDF funkcí slouží uniformní proměnná, která je do shaderu předávána

```
void main(void){
  const vec3 normal = vec3(0, 0, 1);

  const vec3 normPos = normalize(in_Position);
  const vec3 normIncident = normalize(u_incidentVector);

  float brdf = BRDF(normIncident, normPos, normal);
  const vec3 newPos = normPos * brdf;
  gl_Position = u_projMat * u_viewMat * u_modelMat * vec4(newPos, 1.0);
}
```

Výpis 1: Zjednodušený vertex shader

z programu. Parametry jednotlivých funkcí jsou také předávány pomocí uniformních proměnných. Po zpracování ve vertex shaderu je následně vizualizace zpracována fragment shaderem, kde je každý bod vizualizace jednoduše obarven podle vzdálenosti bodu od středu (tzn. dle hodnoty BRDF funkce daného bodu). Obarvení dané hodnoty BRDF funkce je provedeno za pomocí sekvenční barevné mapy „Autumn“ (viz obrázek 5.4). Ukázka výsledné vizualizace je zobrazena na obrázku uživatelského rozhraní aplikace 5.1 v hlavním okně, kde je vizualizována fyzikálně korektní verze BRDF podle Phongova modelu.

Obrázek 5.4: Barevná škála použitá pro obarvení BRDF funkcí

5.2 Vizualizace vzorkování

Každou BRDF funkci je vhodné kombinovat s vhodně zvolenou vzorkovací funkcí. Výběr vzorkovací funkce je důležitý z důvodu optimálního generování vektorů. Na obrázku 5.5 jsou zobrazeny dvojice dvou BRDF funkcí a vzorkovacích funkcí. Z obrázku je vidět, že např. vzorkovací funkce obrázku 5.5d není optimální pro BRDF funkci na obrázku 5.5a. Velké množství vektorů je vygenerováno mimo lalok této BRDF funkce. Opačným extrémem by bylo zvolení kombinace vzorkovací funkce na obrázku 5.5b a BRDF funkce na obrázku 5.5c. V takovém případě vzorkovací funkce nepokryje celou BRDF funkci. Detaily jednotlivých implementovaných vzorkovacích funkcí jsou podrobněji popsány v kapitole 6 Redukce variance Monte Carlo.

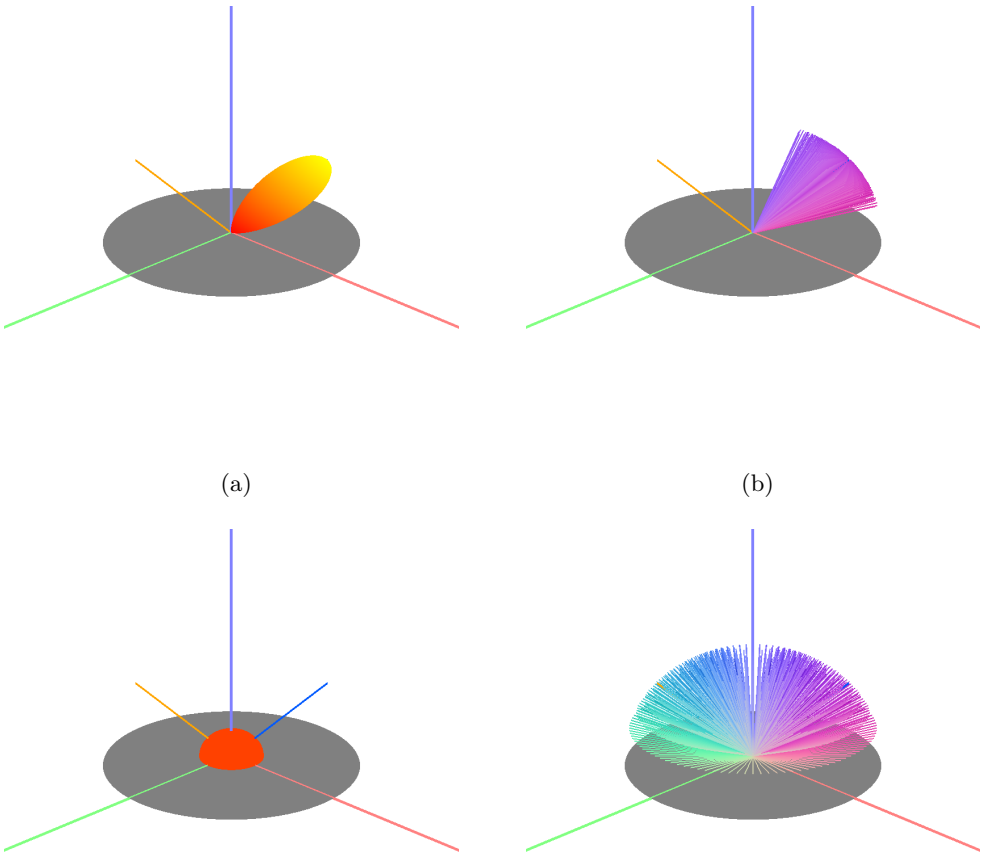
K vizualizaci vzorkovacích funkcí jsem se rozhodl přistoupit pomocí vizualizace vektory. Princip je v zásadě jednoduchý. Vzorkovací funkce pracují tak, že se vygeneruje směr pro náhodně vygenerované hodnoty ξ_1 a ξ_2. Vizualizace potom funguje na principu vygenerování uniformně rozložených hodnot v mřížce s daným rozlišením. Takto generované hodnoty jsou poté převedeny pomocí vzorkovací funkce do vektorů. Uniformně rozložené generované vektory jsou poté přímo zobrazeny, barva každého vektoru přímo koresponduje s jeho směrem. V případě potřeby je možné interaktivně zvýšit nebo snížit počet vizualizovaných vzorků. Ve výchozím nastavení je délka jednotlivých vizualizovaných vektorů jednotková. Je také možné nastavit násobení velikostí vektorů hodnotou pdf. Násobení velikosti vektoru hodnotou pdf demonstruje rozložení pravděpodobnosti vzorku. Čím větší je velikost vzorku, tím větší je jeho pravděpodobnost. Na obrázcích 5.5b a 5.5d jsou zobrazeny ukázky vizualizací vzorkovacích funkcí.

5.3 Vizualizace výsledku path tracingu

Pro ukázku výsledku je vizualizován také výsledný render. Jak již bylo zmíněno, vizualizována je koule, která je osvětlena HDR mapou, nebo konstantním pozadím.

Pro výpočet osvětlení je využita metoda Monte Carlo, v path traceru jsou také implementovány funkce pro optimalizaci výpočtu. Pro ukázku volby vzorkovací funkce využívá path tracer vzorkovací funkci zvolenou v ovládacím panelu. Jednotlivé vzorkovací funkce jsou podrobněji popsány v kapitole 6 Redukce variance Monte Carlo. Implementováno je také vzorkování světelných zdrojů z HDR mapy. Vzorkování světelných zdrojů je možné v ovládacím panelu zapínat a vypínat pro demonstraci efektů jednotlivých typů vzorkování na výsledném renderu.

Kromě analyticky zadané koule je také možné vizualizovat render libovolného načteného objektu. Pro urychlení výpočtu je načtený objekt uložen do BVH stromu – akcelerační struktury, dále popsané v kapitole 6 Redukce variance Monte Carlo. Výsledný render modelu je zobrazen na obrázcích 5.6a a 5.6b.



(a) (b)

(c) (d)

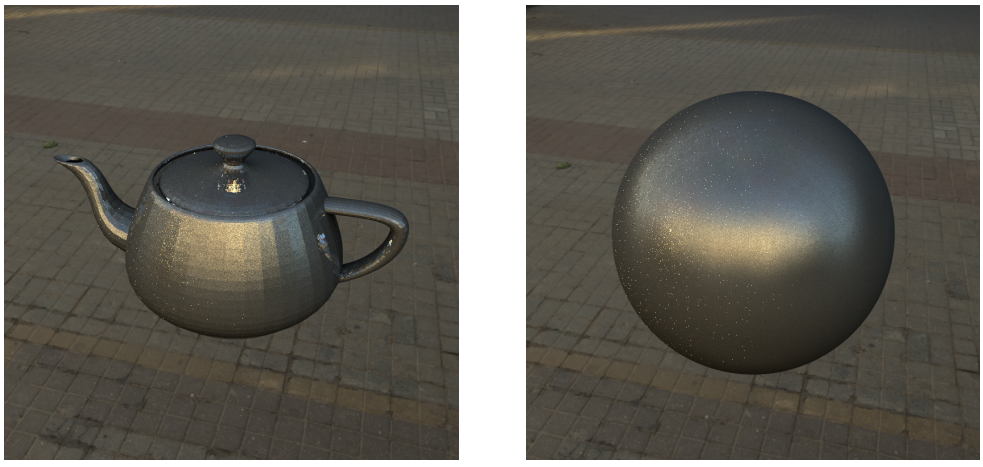
Obrázek 5.5: BRDF funkce a korespondující vzorkovací funkce

```
def trace(ray):
  if sampleLight:
    sampleLight = False
    return sampleRandomLight()
  else
    sampleLight = True
    # Russian roulette
    if currentRecursion > 2 and (albedo * 0.95) <= rng():
    return 0

    if material.isLight():
      return material.emission

    omegaI, pdf = sample(normal)
    brdf = getBrdf(omegaI, normal)
    newHit = generateRay(worldPos, omegaI)
    Li = trace(newHit, currentRecursion + 1)
    return Li * brdf * dot(omegaI, normal) / (pdf * (albedo * 0.95))
```

Výpis 2: Pseudokód pro path tracer



(a) Render načteného objektu (b) Render analyticky zadané koule

Obrázek 5.6: Ukázka vizualizace výsledku path tracingu

5.4 Detaily implementace, použité knihovny a technologie

Aplikace pro vizualizaci je napsána v moderním C++, s použitím standardu C++17. Při implementaci jsem se snažil v maximální možné míře využívat novou funkcionalitu tohoto standardu.

Co se týče architektury mého řešení, je aplikace rozdělena do tří samostatných celků. Hlavní částí aplikace je samotné uživatelské rozhraní, ve kterém jsou všechny ovládací panely. Přímé vykreslování do panelů definovaných knihovnou ImGui není možné, vykreslování jsem tedy prováděl do frame bufferu, resp. do textury. Samotné textury jsem poté zobrazoval v ImGui panelech. OpenGL část aplikace se zabývá samotným vykreslováním BRDF funkcí a vzorkování. Embree část aplikace je zodpovědná za vykreslování objektu za pomocí zvolené BRDF funkce.

Tato kapitola se nejprve věnuje popisu pomocných matematických funkcí, které jsou použity na různých místech aplikace při práci s vektory. Následně se kapitola věnuje popisu knihoven použitých při implementaci aplikace.

5.4.1 Převod vektoru z lokálního do globálního prostoru

V kapitole 6 Redukce variance Monte Carlo jsou popsány metody generování vzorků vektorů používaných pro výpočet odrazu světla. Problémem ale je, že tyto generované vektory nejsou orientovány podél normály. Generované vektory jsou typicky orientovány podle $\hat{up}$ vektoru (vektor určující ve scéně osu směrem nahoru). Pro použití vzorku na povrchu s libovolnou normálou je tedy nutné provést rotaci vygenerovaného vektoru podél normálového vektoru. Tato funkce má také využití např. při generování vzorků pro odrazovou složku BRDF podle Phongova modelu (viz 6.2.2 Vzorkování BRDF podle Phongova modelu). Vzhledem k tomu, že vzorkovací funkce samotná je při renderingu volána velmi často, je vhodné mít všechny kroky vzorkování co nejlépe optimalizované. Jednou z nejlepších metod pro rotaci vektoru je použití transformační matice. Pro rotaci ve 3D prostoru se využívá trojice elementárních rotačních matic, definovaných vzorci 5.1, 5.2 a 5.3 [28]

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix} \tag{5.1}$$

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{5.2}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

Cílem je získání rotační matice pro rotaci vektoru f na vektor t. Po aplikaci rotační matice na vygenerované vektory vzorkovacími funkcemi je dosaženo správné orientace vzorkovaných vektorů. Výpočet rotační matice před dalšími úpravami je definován vzorcem 5.4 [29].

$$\begin{pmatrix} u_x^2 + (1 - u_x^2)\cos\theta & u_x u_y\,(1 - \cos\theta) - y_z \sin\theta & u_x u_z + u_y \sin\theta \\ u_x u_y\,(1 - \cos\theta) + u_z \sin\theta & u_y^2 + \left(1 - u_y^2\right)\cos\theta & u_y u_z\,(1 - \cos\theta) - u_x \sin\theta \\ u_x u_z\,(1 - \cos\theta) - u_y \sin\theta & u_y u_z\,(1 - \cos\theta) + u_x \sin\theta & u_z^2 + (1 - u_z^2)\cos\theta \end{pmatrix} \tag{5.4}$$

Pro výpočet se používá parametr u, což je normalizovaný vektor kolmý na vektory f a t. Z rovnice je také možné odstranit výpočet goniometrických funkcí, vzhledem k tomu, že platí: $\cos\theta = f \cdot t$ a $\sin\theta = \|f \times t\|$. Pro zjednodušení značení je možné provést substituce 5.5.

$$\begin{aligned} c &= f \cdot t \\ v &= f \times t \\ h &= \frac{1 - c}{1 - c^2} = \frac{1 - c}{v \cdot v} \end{aligned}$$

Po aplikaci substitucí je vzorec 5.4 zjednodušen na vzorec 5.5 [29] pro efektivní výpočet bez goniometrických funkcí.

$$R\,(f, t) = \begin{pmatrix} c + h v_x^2 & h v_x v_y - v_z & h v_x v_z + v_y \\ h v_x v_y + v_z & c + h v_y^2 & h v_y v_z - v_x \\ h v_x v_z - v_y & h v_y v_z + v_x & c + h v_z^2 \end{pmatrix} \tag{5.5}$$

5.4.2 Převod sférických souřadnic

Při výpočtu vzorkování se často počítá se sférickými souřadnicemi. Pro převod vektoru definovaného sférickými souřadnicemi do kartézských souřadnic je možné použít vzorec 5.6

$$\begin{aligned} x &= \sin(\theta)\cos(\phi) \\ y &= \sin(\theta)\sin(\phi) \\ z &= \cos(\theta) \end{aligned} \tag{5.6}$$

Zpětný převod normalizovaného vektoru z kartézských souřadnic je také možný pomocí vzorce 5.7. Pro nevalidní hodnoty výpočtu (které nejsou definovány pro funkci arctan) jsou hodnoty θ a ϕ nastaveny na hodnotu 0.

$$\begin{aligned} \theta &= \arctan \frac{\sqrt{x^2 + y^2}}{z} \\ \phi &= \arctan \frac{y}{x} \end{aligned} \tag{5.7}$$

5.4.3 Použité knihovny

Následující kapitoly se věnují popisu knihoven, které jsem použil při implementaci své diplomové práce. Kromě popisu daných knihoven jsem také zdůvodnil použití těchto knihoven.

OpenGL

OpenGL je standardní rozhraní pro vykreslování počítačové grafiky. OpenGL je multiplatformní, je možné s ním pracovat pomocí množství programovacích jazyků a má velmi kvalitní dokumentaci. Jeho nevýhodou může být jeho neobjektový přístup. Práce s OpenGL je v podstatě globální. V současné době je technologie OpenGL nahrazována novějším rozhraním Vulkan, které má oproti OpenGL blíže k samotnému hardwaru. Oproti OpenGL také např. podporuje paralelismus, současná verze už v základu podporuje ray tracing. Pro zpracování této práce byla zvoleno rozhraní OpenGL z důvodu, že nebyly potřeba nové funkce z Vulkanu.

Glad

Jak už bylo zmíněno v předchozím odstavci, OpenGL je v podstatě pouze rozhraní, standard. Je teda potřeba používat nějakou konkrétní implementaci tohoto rozhraní pro konkrétní hardware a konkrétní operační systém. Za tuto implementaci jsou zodpovědné přímo ovladače grafické karty. Pro Windows jsou primárně ovladače dodávány přímo výrobcem grafické karty. Na Linuxu jsou kromě proprietárních ovladačů od výrobců dostupné i komunitní, open-source ovladače. Vzhledem k velké diverzitě všech těchto možností, kdy v době kompilace není známa přesná kombinace, je třeba tuto skutečnost nějakým způsobem ošetřit v kódu.

Pro zjednodušení práce s OpenGL existuje množství knihoven, které zjednodušují přístup k funkcím OpenGL. Pro svou diplomovou práci jsem zvolil knihovnu Glad [30], která je multiplatformní a je distribuována s open-source licencí. Konfigurace této knihovny je poněkud nestandardní, Glad poskytuje webovou službu, kde je možné nakonfigurovat konkrétní požadovanou verzi a seznam požadovaných rozšíření OpenGL. Po konfiguraci je možné stáhnout generované zdrojové soubory. Kromě možnosti generování souborů pomocí webové služby je také možné používat knihovnu jako submodul pro CMake. V takovém případě jsou v konfigurační fázi automaticky staženy generované soubory podle daného nastavení.

Glfw

Kromě samotného rozhraní pro práci s OpenGL je také potřeba nějakým způsobem zobrazit výstup uživateli. K tomu je potřeba vytvořit okno. Pro práci s okny existuje množství knihoven, pro tuto diplomovou práci byla vybrána knihovna Glfw [31]. Kromě tvorby a práce s okny tato knihovna také podporuje práci s uživatelskými vstupy.

Výhodou knihovny Glfw je primárně odlehčenost celé knihovny. Glfw je také šířena s open-source licencí. Glfw je multiplatformní, s podporou pro všechny běžné operační systémy.

Dear ImGui

Jak již bylo zmíněno, knihovna Glfw poskytuje pouze rozhraní pro práci s okny. To znamená, že nepodporuje samotnou tvorbu uživatelského rozhraní. Pro tvorbu uživatelských rozhraní je teda potřeba použít jinou knihovnu.

Pro tvorbu uživatelských rozhraní existuje pro jazyk C++ velké množství různých knihoven. V různých linuxových distribucích jsou v podstatě standardem primárně dvě knihovny: QT a GTK. Obě knihovny jsou multiplatformní, ale ani jedna není úplně vhodná pro mé použití.

Knihovna QT není příliš vhodná z důvodu velikosti této knihovny. QT v zásadě není jen knihovna specializovaná pouze na uživatelská rozhraní, ale je to velmi komplexní a obsáhlý framework. Pro velmi jednoduchou aplikaci, jako je aplikace implementovaná v této diplomové práci by byla většina funkcionality zbytečná. Knihovna QT také nepřináší žádnou výhodu oproti použitému řešení.

Oproti tomu knihovna GTK je knihovna, která se primárně zabývá tvorbou uživatelských rozhraní. Je však primárně vyvíjena pro použití na Linuxu, a přestože je multiplatformní a je možné ji použít na Windows, použití na Windows není primárně bráno v potaz. Knihovna GTK je také poněkud složitější na používání.

Po zhodnocení možností jsem se nakonec rozhodl použít knihovnu Dear ImGui [32]. Výhodou použití této knihovny je primárně její jednoduchost použití. Do již existujícího kódu je velmi snadno integrovatelná. Samotná tvorba uživatelského rozhraní je intuitivní a přímočará. Neexistuje oficiálně žádný grafický editor (přestože existují komunitní editory), ale knihovna je pro používání natolik jednoduchá, že takový editor ani není potřeba. Dear ImGui je možné používat nejen s OpenGL, ale i s jinými rozhraními pro počítačovou grafiku (mimo jiné kromě OpenGL oficiálně podporuje Vulkan, Metal, nebo DirectX). Dear ImGui také poskytuje množství ukázkových programů, které jsou vytvořeny pro různé kombinace platforem, použitých technologií a knihoven, ze kterých je možné vycházet.

Knihovna Dear ImGui je distribuována s open-source licencí a je multiplatformní. Kromě základní verze této knihovny existuje také separátní verze s podporou pro „docking“ jednotlivých panelů do pracovní plochy aplikace. Pro svou diplomovou práci jsem zvolil právě tuto verzi knihovny.

Glm

Obecně v počítačové grafice se velmi často pracuje s vektory, maticemi a operacemi s nimi. Z hlediska výkonu je kritické, aby vektorové a maticové operace byly co nejvíce optimalizované. Z tohoto důvodu jsem se rozhodl pro práci s vektory a maticemi použít knihovnu Glm [33]. Knihovna Glm se velmi jednoduše používá, je distribuována pod open-source licencí a je multiplatformní. Knihovna Glm je implementována tak, aby splňovala standard jazyka GLSL, používaném v OpenGL shaderech. Není však na OpenGL závislá a lze tak tuto knihovnu použít i pro jiné účely než jen pro počítačovou grafiku.

Assimp

Aplikace má možnost načítat uživatelské modely pro náhled renderu a nahradit tak výchozí analyticky zadaný model koule. Pro samotné načítání modelů jsem použil knihovnu Assimp [34]. Knihovna Assimp je distribuována s 3-bodovou BSD licencí.

Pro zjednodušení je ve výchozím stavu vypnuta většina funkcionality této knihovny, podporováno jsem ve výchozím stavu nechal pouze načítání modelů ve formátu Wavefront obj. Vypnutá funkcionalita lze jednoduše opět zapnout při konfiguraci CMake projektu. Kromě načtení modelu se načítají také materiály. Načtené materiály ale nejsou použity, protože aplikace pracuje s jedním globálním materiálem.

FreeImage

Aplikace umožňuje práci s obrázky – načítání textur a ukládání snímků z vizualizace BRDF funkce i vizualizace finálního renderu. K tomuto účelu jsem použil rozhraní knihovny FreeImage [35].

Embree

Jelikož součástí aplikace je i náhled na výsledný render pomocí metody path tracing, použil jsem pro zjednodušení práce knihovnu Embree [36], která se stará o tvorbu samotných paprsků a následnou traverzaci paprsku a scény. Jelikož primárně je renderován analyticky zadaný model koule, implementoval jsem nad knihovnou Embree i drobné rozhraní pro použití jak načtených modelů, tak analytických těles. Analyticky zadaná tělesa i načtené modely je tak možné používat v jedné scéně.

Knihovna Embree je spravována organizací Intel, je multiplatformní a distribuována s open-source licencí. Knihovna je optimalizována pro použití s Intel procesory. Nejlepšího výkonu je možné dosáhnout s procesory podporujícími AVX instrukce a při použití Intel kompilátoru.

Pro path tracing je možné využívat spoustu jiných knihoven s výpočtem, jak na CPU, tak na GPU. Je možné používat např. přímo Vulkan pro použití na grafické kartě, případně Nvidia OptiX s výpočty také na grafické kartě. Rozhodl jsem se pro Embree z důvodu nižších systémových

požadavků. Výsledná aplikace je tedy funkční i na méně výkonných počítačích. Zároveň path tracing není primární funkcionalita mé diplomové práce a lze tak použít radši řešení s vyšší kompatibilitou, než s vyšším výkonem (za cenu nižší kompatibility).

Spdlog

Při vývoji aplikace je velmi užitečné používat různé informační výpisy. Informační výpisy mohou mít různé úrovně sdělení (např. informace, varování, chyby…). Přestože implementovat takové výpisy na straně aplikace by nebylo složité, rozhodl jsem se pro knihovnu Spdlog [37]. Tato knihovna velmi usnadňuje práci s výpisy, kdy je možné jednoduše nastavit úroveň výpisů, které jsou aktuálně zobrazovány (tzn. např. lze zapnout pouze vypisování varování a chyb a vypnout informační výpisy). Také lze velmi jednoduše vypisovat nejen do konzole, ale i do souboru. Výhodou je také moderní přístup k výpisům, kdy je pro formátování výstupu použita knihovna fmt, která byla převzata do standardu C++20. Knihovna také podporuje výpisy z různých vláken v programu, nebo asynchronní přístup. Knihovna spdlog je multiplatformní a je distribuována s open-source licencí.

Kapitola 6

Optimalizace Monte Carlo

Následující kapitoly se zabývají různými způsoby optimalizace řešení zobrazovací rovnice pomocí metody Monte Carlo.

6.1 Optimalizace datových struktur

Způsob uložení objektů v datových strukturách má markantní vliv na výkon celé aplikace. Z tohoto důvodu je při implementaci vhodné pro uložení dat použít akcelerační struktury. Akcelerační struktura je používána pro minimalizaci počtu objektů, které je při traverzaci paprsku scénou potřeba vzít v úvahu. Akcelerační struktury je možné rozdělit do dvou kategorií:

1. Struktury dělící objekty

2. Struktury dělící prostor

Struktury dělící objekty shlukují objekty umístěné ve scéně u sebe. Typickým příkladem této kategorie je např. BVH strom. Struktury dělící prostor rozdělují prostor na menší části, které obsahují tělesa. Typickým příkladem této kategorie je např. kdTree.

6.1.1 Naivní uložení objektů v poli

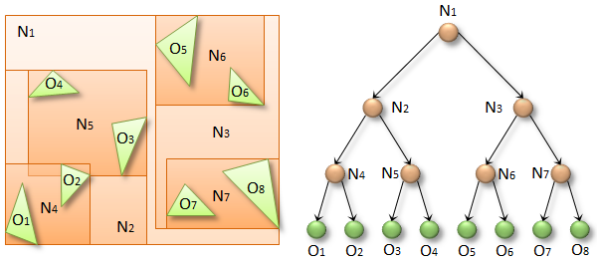
Jednoduché uložení objektů v poli je nejjednodušší způsob pro řešení tohoto problému. Následná traverzace paprsku scénou se při profilingu ukáže jako jedna z nejnáročnějších částí aplikace, co se výpočetního času týče. Tento problém narůstá s počtem objektů ve scéně, jelikož je vždy nutné pro každý paprsek ve scéně vzít v úvahu každý objekt v poli, z toho vyplývá výpočetní složitost $\mathcal{O}(n)$, kde n je počet polygonů ve scéně. Uvádí se, že až 95% výpočetního času programu zabere samotná traverzace [38]. Tento způsob je vhodný pouze pro malé scény s nízkým počtem objektů, případně pro prvotní rychlé a jednoduché dosažní výsledků.

6.1.2 Bounding Volume Hierarchy

BVH (Bounding Volume Hierarchy) je způsob rozdělení těles hierarchicky, do stromové struktury. Stromová struktura je použita pro optimalizaci traverzace, výpočetní složitost $\mathcal{O}(\log n)$ vycházející z průchodu stromem není garantována, ale je možné se k ní přiblížit[38].

Pro reprezentaci se nejčastěji používá binární strom. Uzly ve stromu reprezentují obalovou strukturu definující objem, ve kterém jsou uloženy další uzly.

Listy stromu reprezentují přímo tělesa, každé těleso může být ve stromu právě jednou. Těleso listu může být jak jednotlivý objekt ve scéně, tak případně jednotlivý, samostatný polygon objektu. Ukázka reprezentace objektů v BVH stromu je zobrazena na obrázku 6.1



Obrázek 6.1: Ukázka BVH stromu [39]

Obalová struktura

Obalová struktura může být libovolné 3D těleso, které definuje hranice objektů obsažených v obalové struktuře. Obalová struktura by měla v ideálním případě vytvořit takové těleso, které co nejtěsněji obalí objekty. Co nejpřesněji vytvořená obalová struktura se používá např. při detekci kolizí. Pro potřebu urychlení traverzace paprsku scénou je ale možné použít i jednodušší struktury. Jednou z nejčastěji používaných jednodušších struktur je struktura AABB.

AABB (Axis-Aligned Bounding Box) je struktura, definována dvěma body. Tyto dva body definují rohy pomyslného kvádru. Tento kvádr ve scéně definuje hranice objektů náležících tomuto objemu. Použití AABB struktury pro urychlení výpočtu traverzace paprsku je výhodné z důvodu triviálního výpočtu průniku paprsku s touto strukturou.

Tvorba BVH stromu

Pro tvorbu BHV stromu existuje množství různých přístupů. Tvorba BVH stromu v zásadě probíhá rekurzivním dělením prostoru podle nějaké metriky. Prostor je definován obalovou strukturou, která se rekurzivně dělí tak dlouho, dokud listy BVH stromu nejsou jednotlivá tělesa.

Jednou z možností je dělení prostoru na poloviny jednoduchým dělením struktury na dvě části v geometrické polovině prostoru. Dělení se vždy provádí pouze v jedné ose. Osa pro dělení je zvolena

ta, ve které je velikost aktuální obalové struktury maximální. Tato metoda není příliš efektivní, jelikož může jednoduše nastat situace, kdy není BVH strom vyvážený. Průchod nevyváženým stromem pak není efektivní.

Další z možností je dělení prostoru na poloviny takovým způsobem, aby v každé ze dvou částí byl stejný počet objektů. Tímto je vyřešen nedostatek primitivního dělení. V takovém případě se tělesa v aktuálně prohledávané obalové struktuře seřadí do pole podle pozice a vybere se těleso v polovině pole. Následně se vytvoří dvě nové obalové struktury, každá obsahující tělesa z jedné poloviny pole.

Jedním z aktuálně nejpoužívanějších způsobů pro dělení obalové struktury je metoda SAH (Surface Area Heuristic). Tato metoda používá jako svou metriku minimalizaci obsahů jednotlivých podstromů. To znamená, že při použití SAH je součet obsahů dvou nově vytvořených obalových struktur v aktuální obalové struktuře minimální. Při použití této metody je při dělení potřeba několikrát rozdělit obalovou strukturu a porovnat jednotlivé možnosti. Z tohoto důvodu je tvorba BVH stromu s použitím SAH výpočetně náročnější než první dvě možnosti. Díky kvalitnějšímu rozdělení prostoru je ale použití SAH výhodnější. V porovnání s první a druhou popisovanou metodou je traverzace při použití SAH až 2× rychlejší [40]. Metoda SAH je ve výchozím stavu použita při tvorbě scény za pomocí knihovny Embree.

6.1.3 Kd-tree

Kd-tree je metoda pro dělení prostoru pomocí tzv. BSP (Binary Space Partitioning). Kd-tree dělí prostor na oblasti, které nemusí mít stejnou velikost. Zároveň lze dělení provést podél libovolného směru, směr nemusí být rovnoběžný s osou souřadného systému. Tato vlastnost ale může být pro potřeby urychlení traverzace paprsku kontraproduktivní. Proto je vhodnější při použití Kd-tree vždy dělit prostor podél jedné z os souřadného systému.

Stejně jako BVH strom je Kd-tree reprezentován binárním stromem. Na rozdíl od BVH, v Kd-tree se může objekt vyskytovat vícekrát. Jednotlivé obalové struktury se tak mohou překrývat.

Alternativní variantou je dále datová struktura Octree. Octree je v zásadě Kd-tree, který však nepoužívá pro dělení jeden řez, ale tři řezy. To znamená, že každý list stromové struktury má osm potomků. Hlavní výhodou použití Octree místo Kd-tree je jednodušší a rychlejší vkládání nových prvků.

Při tvorbě Kd-tree i Octree lze použít při dělení stejné metriky, jako u BVH. Na rozdíl od BVH je však metrika SAH upravena. V tomto případě SAH metrika upřednostňuje řez, při kterém vznikne potomek, který je zároveň list.

6.2 Optimalizace vzorkování

Jak již bylo demonstrováno dříve na praktickém příkladu v kapitole 5.2, je vhodné kombinovat zvolenou BRDF funkci s funkcí pro vzorkování. Následující kapitoly se zabývají rozborem vzorkovacích funkcí, které lze použít v kombinaci s BRDF funkcemi popisovanými v této práci.

6.2.1 Vzorkování hemisféry

Jednou z nejjednodušších metod pro vzorkování je vzorkování hemisféry. Uniformní vzorkování hemisféry je definováno vzorcem 6.1 (ve sférických souřadnicích). Rozdělení pravděpodobnosti je konstantní, definováno vzorcem 6.2. Při tomto vzorkování jsou všechny paprsky rovnoměrně rozdělené, všechny směry mají stejnou pravděpodobnost odrazu. Takové vzorkování je možné použít pro všechny BRDF funkce. Nevýhodou tohoto vzorkování je ale neoptimálnost vůči různým BRDF funkcím. Ideálně by vzorkovací funkce měla co nejpřesněji kopírovat BRDF funkci.

$$\begin{aligned} \theta &= \arccos(\xi_1) \\ \phi &= 2\pi\xi_2 \end{aligned} \tag{6.1}$$

$$p = \frac{1}{2\pi} \tag{6.2}$$

Jedna z možností optimalizace je použití vzorkování hemisféry závislého na cosinu úhlu mezi vstupním paprskem a normálou povrchu. Základní tezí je, že paprsky s největším úhlem mají nejmenší vliv na výsledné osvětlení v daném bodě. Vzorkování by tedy mělo mít rozložení vzorků více kumulované ve směru normály oproti ostatním směrům. Jednou z možností, jak generovat paprsky touto metodou je pomocí Malleyho metody. Tato metoda spočívá v generování bodu v jednotkovém disku pomocí soustředných kruhů [41]. Výsledné souřadnice v kartézském systému se vypočítají pomocí vzorce 6.3. Pro výpočet je potřeba převést ξ_1 a ξ_2 z intervalu $\langle 0, 1)$ do intervalu $\langle -1, 1)$ Rozdělení pravděpodobnosti je definováno vzorcem 6.4.

$$\begin{aligned} r &= \begin{cases} \xi_1 & |\xi_1| > |\xi_2| \\ \xi_2 & jinak \end{cases} \\ \Theta &= \begin{cases} \frac{\pi}{4}\frac{\xi_2}{\xi_1} & |\xi_1| > |\xi_2| \\ \frac{\pi}{2} - \frac{\pi}{4}\frac{\xi_1}{\xi_2} & jinak \end{cases} \\ x &= r\cos(\Theta) \\ y &= r\sin(\Theta) \\ z &= \sqrt{\max(0, 1 - x^2 + y^2)} \end{aligned} \tag{6.3}$$

$$p = \frac{\cos\theta}{\pi} \tag{6.4}$$

6.2.2 Vzorkování BRDF podle Phongova modelu

Pro vzorkování BRDF podle Phongova modelu je potřeba rozhodnout, kterou část BRDF vzorkujeme – jestli difuzivní nebo odrazovou. Toto rozhodnutí je možné vytvořit náhodně, pomocí vygenerování náhodného čísla ξ z intervalu $\langle 0, k_d + k_s \rangle$. Pokud $\xi < k_d$, je vzorkována difuzivní část. V opačném případě je vzorkována odrazová část [42].

6.2.3 Vzorkování difuzivní části

Difuzivní část je možné vzorkovat pomocí hemisféry (6.2.1). Je možné také použít hemisféru závislou na cosinu úhlu mezi vstupním paprskem a normálou povrchu.

Vzorkování odrazové části

Pro vzorkování odrazové části je vzorkován Phongův cosinový lalok, který je vycentrován okolo ω_r. Výpočet vzorkování odrazové části je definován vzorcem 6.5 [42]. Rozdělení pravděpodobnosti je definováno vzorcem 6.6. Výsledek tohoto vzorkování je vycentrován okolo normály, je tedy potřeba rotovat výsledek podél ω_r. K rotaci vektoru je možné využít vzorec 5.5.

$$\begin{aligned} \theta &= \arccos(\xi_1^{\frac{1}{n+1}}) \\ \phi &= 2\pi\xi_2 \end{aligned} \tag{6.5}$$

$$p = \frac{n+1}{2\pi}\cos^n\theta \tag{6.6}$$

Výsledné vzorkování

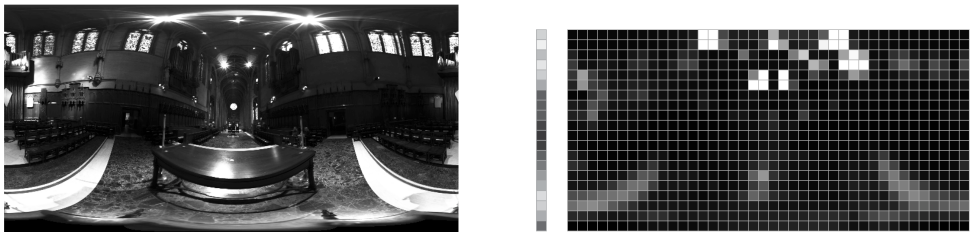
Po spojení vzorkování odrazové a difuzivní části je vhodné adaptovat i výpočet samotného BRDF. Při vzorkování difuzivní části je třeba jako BRDF funkci použít pouze difuzivní část BRDF podle Phongova modelu, při vzorkování odrazové části je použita odrazová část BRDF podle Phongova modelu. Pokud by se použilo BRDF podle Phongova modelu bez úpravy, nebylo by dosaženo snížení variance ve výsledném obrazu. [42]

6.2.4 Vzorkování světelných zdrojů z HDR mapy

Pro vzorkování z HDR mapy je potřeba nejprve určit metodiku vzorkování z diskrétní 2D veličiny. Je možné interpretovat 2D veličinu jako 1D veličinu, ale takový přístup není optimální. Lepší ře-

šení je přijít se vzorkováním, které vzorkuje 2D veličinu pomocí dvou náhodných čísel, pomocí dvou marginálních distribucí. V takovém případě je možné vzorkovat z 2D veličiny jako z matice. Nejprve je potřeba vypočítat marginální distribuce. Je možné počítat distribuce ve sloupcích nebo v řádcích. V případě výpočtu marginálních distribucí ve sloupcích je potom vzorkován řádek z daného sloupce, v opačném případě je vzorkován sloupec z daného řádku. Pomocí hodnot ξ_1 a ξ_2 je vybrána distribuce, resp. výsledná vzorkovaná hodnota. [41]

Tento postup je vhodný provádět na obrazových datech, na kterých je sníženo rozlišení. Díky snížení rozlišení se zjednoduší výpočet a spojí se větší světelné zdroje do jedné oblasti.



(a) HDR mapa (b) HDR mapa s marginální distribucí

Obrázek 6.2: Postup vzorkování světelných zdrojů [41]

6.2.5 Multiple importance sampling

Jednotlivé, samostatné možnosti vzorkování mají pozitivní vliv na výkon aplikace. Samostatné vzorkování pouze BRDF funkce, nebo pouze světelných zdrojů ale nedává optimální výsledky (viz obrázek 6.3a a obrázek 6.3b). Ve článku [43] je představena metoda MIS (Multiple Importance Sampling). Pomocí metody MIS lze spojit libovolný počet vzorkovacích technik. Pro výsledné vzorkování je pak použit výpočet definovaný vzorcem 6.7. Pro výpočet je nutné každé vzorkovací technice přiřadit koeficient w. Tento koeficient nemusí být konstantní, je možné jej dynamicky měnit podle současného vzorku. Pro všechny koeficienty w pak musí platit podmínky 6.8 a 6.9.

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})} \tag{6.7}$$

$$\sum_{i=1}^{n} w_i(x) = 1 \text{ pro } f(x) \neq 0 \tag{6.8}$$

$$w_i(x) = 0 \text{ pro } p_i(x) = 0 \tag{6.9}$$

Pro získání koeficientů w je ve článku [43] navržena dvojice heuristik, která je definovaná vzorcem 6.10. Jedná se o tzv. „balance heuristic“ a „power heuristic“. Tato dvojice heuristik patří aktuálně mezi jedny z nejpoužívanějších.

O balance heuristic – vyváženou heuristiku se jedná v případě, kdy $\beta = 1$. Ve článku [43] je tvrzení, že žádná jiná heuristika (v té době) neměla výrazně lepší výsledky než právě tato heuristika. V případě, že $\beta > 1$, jedná se o Power heuristic. Tato heuristika je vhodná pro aplikace s obecně menší variancí. Parametr β je možné nastavit libovolně, experimentálně, ve článku [43] je navrženo nastavení $\beta = 2$. Výsledek použití metody MIS s Balance heuristikou je zobrazen na obrázku 6.3c

$$w_i\left(x\right) = \frac{\left[n_i p_i(x)\right]^\beta}{\sum_k \left[n_k p_k(x)\right]^\beta} \tag{6.10}$$

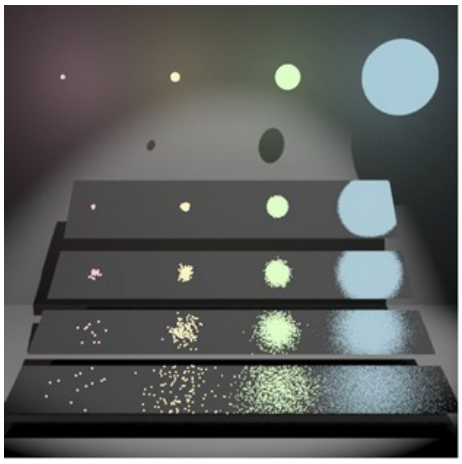
Balance a Power heuristika nabízí robustní kombinaci vzorkování, kdy i při horší kombinaci vstupů není negativně ovlivněn celkový výstup. Nevýhodou ale je horší efektivita tohoto přístupu. V některých případech může např. dojít k situaci, kdy by lepšího výsledku bylo dosaženou použitím pouze jedné vzorkovací techniky. Tento problém a jeho následné řešení byl podrobně popsán v článku [44]. Výpočet heuristiky je popsán vzorcem 6.11. Heuristika pro svůj výpočet používá dodatečný parametr α, který lze získat pomocí řešení lineárního systému 6.12. Pro výpočet je nutná matice hodnot a a vektor hodnot b. Tyto hodnoty jsou získány pomocí rovnic 6.13.

$$w_i\left(x\right) = \frac{\alpha_i p_i\left(x\right)}{f\left(x\right)} + \frac{n_i p_i(x)}{\sum_{k=1}^{N} n_k p_k(x)} \left(1 - \frac{\sum_{k=1}^{N} \alpha_k p_k\left(x\right)}{f(x)}\right) \tag{6.11}$$

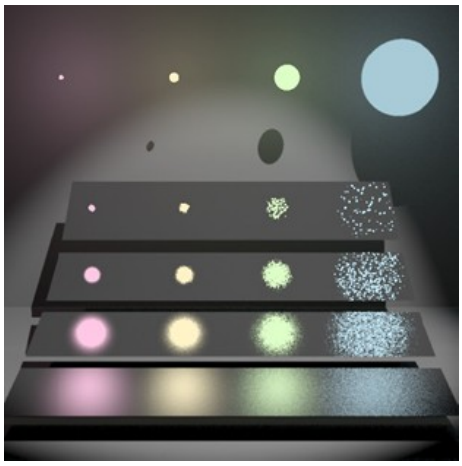
$$\begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_N \end{pmatrix} \tag{6.12}$$

$$a_{ij} = \int \frac{p_i p_j}{\sum_{k=1}^{N} n_k p_k} \tag{6.13}$$

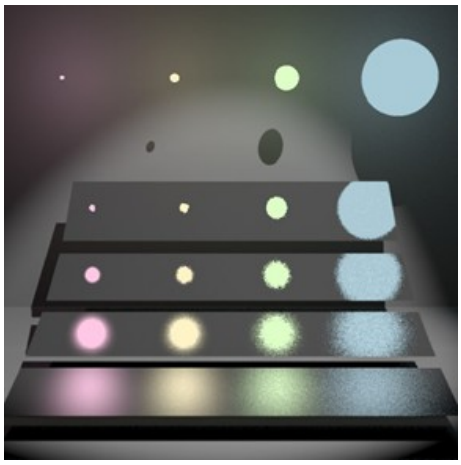
$$b_i = \int \frac{p_i f}{\sum_{k=1}^{N} n_k p_k} \tag{6.14}$$



(a) Vzorkování BRDF



(b) Vzorkování světelných zdrojů



(c) Multiple importance sampling

Obrázek 6.3: Srovnání výsledků vzorkování [43]

Kapitola 7

Závěr

Byla vytvořena multiplatformní aplikace v jazyce C++, která umožňuje interaktivně vizualizovat vybrané analyticky zadané BRDF funkce i s výsledným renderem. V aplikaci je také možné vizualizovat vzorkovací funkce a demonstrovat vliv výběru vzorkovacích funkcí na výsledný render. Aplikace je snadno rozšiřitelná o případné další BRDF funkce a vzorkovací funkce.

V první kapitole bylo sjednoceno značení veličin a parametrů BRDF funkcí, které byly používány ve vzorcích. U některých veličin byl také upřesněn výpočet, který je možné použít pro získání jejich hodnoty. U parametrů BRDF funkcí bylo také zmíněno, který parametr je pro kterou BRDF funkci relevantní.

Ve druhé kapitole diplomové práce byly zpracovány existující aplikace, sloužící k vizualizaci a návrhu BRDF funkcí. Popsány byly funkcionality jednotlivých aplikací, byly zmíněny zajímavé vlastnosti a nedostatky. Na konci druhé kapitoly byly popsány požadavky na implementaci aplikace, vycházející z vlastností existujících aplikací a řešící některé z nedostatků.

Ve třetí kapitole diplomové práce byly teoreticky popsány principy osvětlení scény společně s osvětlením pomocí HDR mapy. Byl popsán princip sledování paprsku, zobrazovací rovnice a její řešení s použitím metody Monte Carlo.

Ve čtvrté kapitole byl teoreticky rozebrán princip BRDF funkce, společně s podmínkami pro fyzikální přesnost. Následně byly popsány vybrané BRDF funkce. Každá BRDF funkce byla popsána analyticky, byly popsány případné parametry a také bylo uvedeno, pro které materiály se daná BRDF funkce hodí. Také byla každá BRDF funkce vizualizována pomocí snímku z vytvořené aplikace. Na konci kapitoly byl pak srovnán výkon jednotlivých BRDF funkcí za pomocí naměřených dat.

Pátá kapitola se věnovala popisu vytvořené aplikace. Jednotlivé části aplikace byly popsány a výsledky těchto částí byly zobrazeny. Byl rozebrán princip zobrazení BRDF funkcí, vybraného vzorkování a také výsledku path tracingu vzniklého pomocí metody path tracing. Dále byly také zpracovány důležité pomocné matematické funkce. Nakonec byly popsány použité knihovny.

V poslední kapitole byly rozebrány jednotlivé možnosti optimalizace metody Monte Carlo pro řešení zobrazovací rovnice. Byly teoreticky popsány akcelerační datové struktury a podrobněji bylo rozvedeno použití BVH stromu a Kd-tree. Následně byly teoreticky zpracovány možnosti optimalizace pomocí vzorkování. Bylo popsáno vzorkování samotných BRDF funkcí i vzorkování světelných zdrojů z HDR mapy. Následně bylo popsáno možné spojení těchto dvou vzorkovacích metod pomocí metody Multiple Importance Sampling.

Možná vylepšení aplikace

Aplikace jako taková by mohla být dále rozšířena o několik různých funkcí. V aktuálním stavu je aplikace závislá na BRDF funkcích přímo implementovaných v kódu. Bylo by teda možné rozšířit aplikaci o načítání uživatelsky definovaných BRDF funkcí. Zde však vzniká problém, jelikož je potřeba definovat BRDF funkci pro GLSL shader a také pro výpočet pomocí Embree. Bylo by teda potřeba tento výpočet sjednotit. Možným řešením by bylo použití např. jazyka Lua pro definici BRDF funkcí a jejich parametrů. Kód by pak musel být zpracován a zkompilován za běhu aplikace pro glsl shader. Obohacující by také mohla být podpora pro načítání BRDF modelů získaných z reálně naměřených dat. Nabízelo by se tak použití např. dat z databáze MERL. Další možností by mohla být interaktivní úprava kódu pro BRDF funkci přímo za běhu aplikace. Funkce by mohlo být možné upravovat přímo textově, nebo by mohl být vytvořen systém pro vizuální programování. Takové systémy se stávají populární, lze je nalézt např. v Unreal Engine 4, nebo v Blender. Z hlediska výkonu by aplikace mohla být upravena tak, aby byl volitelný způsob výpočtu finálního renderu. Kromě výpočtu na CPU s použitím knihovny Embree by mohla být přepínatelná podpora pro výpočet na GPU např. pomocí technologie Vulkan, Nvidia OptiX, nebo OpenCL.

Literatura

1. BURLEY, Brent. Physically-Based Shading at Disney. In: 2012. Dostupné také z: https://media.disneyanimation.com/uploads/production/publication_asset/48/asset/s2012_pbs_disney_brdf_notes_v3.pdf.
2. STUDIOS, Walt Disney Animation. *Disney BRDF Explorer* [https://github.com/wdas/brdf]. GitHub, [b.r.].
3. WALKER, Peter. A VISUALIZATION SYSTEM FOR BIDIRECTIONAL REFLECTANCE DISTRIBUTION FUNCTIONS. 2021-03. Dostupné také z: https://www-users.cs.umn.edu/~gmeyer/papers/walker-thesis-1999.pdf.
4. FORÉS, A.; PATTANAIK, S.; BOSCH, C.; PUEYO, X. BRDFLab: A general system for designing BRDFs. In: *CEIG*. 2009.
5. FORÉS, A.; PATTANAIK, S.; BOSCH, C.; PUEYO, X. *BRDFLab: A tool to design, fit and render BRDFs* [http://brdflab.sourceforge.net]. SourceForge, [b.r.].
6. COLBERT, Mark; PATTANAIK, Sumanta; KRIVANEK, Jaroslav. BRDF-Shop: Creating physically correct bidirectional reflectance distribution functions. *IEEE computer graphics and applications*. 2006-01, roč. 26, s. 30–6. Dostupné z DOI: 10.1109/MCG.2006.13.
7. KAJIYA, James T. The Rendering Equation. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1986, s. 143–150. SIGGRAPH '86. ISBN 0897911962. Dostupné z DOI: 10.1145/15922.15902.
8. IMMEL, David S.; COHEN, Michael F.; GREENBERG, Donald P. A Radiosity Method for Non-Diffuse Environments. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1986, s. 133–142. SIGGRAPH '86. ISBN 0897911962. Dostupné z DOI: 10.1145/15922.15901.
9. HAINES, Eric; AKENINE-MÖLLER, Tomas (ed.). *Ray Tracing Gems*. Apress, 2019. http://raytracinggems.com.

10. DUTRÉ, Philip. *Global Illumination Compendium*. Department of Computer Science, 2003. Dostupné také z: https://people.cs.kuleuven.be/~philip.dutre/GI/TotalCompendium.pdf. Katholieke Universiteit Leuven.

11. DEBEVEC, Paul. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1998, s. 189–198. SIGGRAPH '98. ISBN 0897919998. Dostupné z DOI: 10.1145/280814.280864.

12. DŘÍMAL, Jiří; TRUNEC, David; BRABLEC, Antonín. *Úvod do metody Monte Carlo*. Brno: Masarykova univerzita, 2006. Dostupné také z: https://www.physics.muni.cz/~trunec/mc.pdf.

13. PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg. 05 - Color and Radiometry. In: PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg (ed.). *Physically Based Rendering (Third Edition)*. Third Edition. Boston: Morgan Kaufmann, 2017, s. 313–353. ISBN 978-0-12-800645-0. Dostupné z DOI: https://doi.org/10.1016/B978-0-12-800645-0.50005-1.

14. HAPKE, Bruce. A miscellany of bidirectional reflectances and related quantities. In: *Theory of Reflectance and Emittance Spectroscopy*. 2. vyd. Cambridge University Press, 2012, s. 263–286. Dostupné z DOI: 10.1017/CBO9781139025683.010.

15. RADZISZEWSKI, Michal; ALDA, Witold. Family of Energy Conserving Glossy Reflection Models. In: 2008-06, sv. 5102, s. 46–55. Dostupné z DOI: 10.1007/978-3-540-69387-1_6.

16. MONTES SOLDADO, Rosana; UREÑA ALMAGRO, Carlos. *An Overview of BRDF Models*. 2012-03. Dostupné také z: http://hdl.handle.net/10481/19751.

17. KOPPAL, Sanjeev J. Lambertian Reflectance. In: *Computer Vision: A Reference Guide*. Ed. IKEUCHI, Katsushi. Boston, MA: Springer US, 2014, s. 441–443. ISBN 978-0-387-31439-6. Dostupné z DOI: 10.1007/978-0-387-31439-6_534.

18. PHONG, Bui Tuong. Illumination for Computer Generated Pictures. *Commun. ACM*. 1975-06, roč. 18, č. 6, s. 311–317. ISSN 0001-0782. Dostupné z DOI: 10.1145/360825.360839.

19. BLINN, James F. Models of Light Reflection for Computer Synthesized Pictures. In: *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*. San Jose, California: Association for Computing Machinery, 1977, s. 192–198. SIGGRAPH '77. ISBN 9781450373555. Dostupné z DOI: 10.1145/563858.563893.

20. LAFORTUNE, Eric; WILLEMS, Yves. Using the Modied Phong Reflectance Model for Physically Based Rendering. 1998-01.

21. PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg. 08 - Reflection Models. In: PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg (ed.). *Physically Based Rendering (Third Edition)*. Third Edition. Boston: Morgan Kaufmann, 2017, s. 507–568. ISBN 978-0-12-800645-0. Dostupné z DOI: https://doi.org/10.1016/B978-0-12-800645-0.50008-7.

22. SCHLICK, Christophe. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*. 1994. ISSN 1467-8659. Dostupné z DOI: 10.1111/1467-8659.1330233.

23. WALTER, Bruce; MARSCHNER, Stephen R.; LI, Hongsong; TORRANCE, Kenneth E. Microfacet Models for Refraction through Rough Surfaces. In: *Proceedings of the 18th Eurographics Conference on Rendering Techniques*. Grenoble, France: Eurographics Association, 2007, s. 195–206. EGSR'07. ISBN 9783905673524.

24. COOK, Robert; TORRANCE, Kenneth. A Reflectance Model for Computer Graphics. *ACM Trans. Graph.* 1982-01, roč. 1, s. 7–24. Dostupné z DOI: 10.1145/965161.806819.

25. OREN, Michael; NAYAR, Shree K. Generalization of Lambert's Reflectance Model. In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1994, s. 239–246. SIGGRAPH '94. ISBN 0897916670. Dostupné z DOI: 10.1145/192161.192213.

26. TAN, Qinzi. *Mesh Deformation Study with a Sphere*. Medium, 2019-10. Dostupné také z: https://medium.com/@qinzitan/mesh-deformation-study-with-a-sphere-ceee37d47e32.

27. AHN, Song Ho. [B.r.]. Dostupné také z: http://www.songho.ca/opengl/gl_sphere.html.

28. HUGHES, John F.; DAM, Andries van; MCGUIRE, Morgan; SKLAR, David F.; FOLEY, James D.; FEINER, Steven; AKELEY, Kurt. *Computer Graphics: Principles and Practice*. 3. vyd. Upper Saddle River, NJ: Addison-Wesley, 2013. ISBN 978-0-321-39952-6.

29. MÖLLER, Tomas; HUGHES, John F. Efficiently Building a Matrix to Rotate One Vector to Another. *Journal of Graphics Tools*. 1999, roč. 4, č. 4, s. 1–4. Dostupné z DOI: 10.1080/10867651.1999.10487509.

30. HERBERTH, David. *GLAD* [https://github.com/Dav1dde/glad]. GitHub, [b.r.].

31. GLFW. *GLFW* [https://github.com/glfw/glfw]. GitHub, [b.r.].

32. CORNUT, Omar. *Dear ImGui* [https://github.com/ocornut/imgui]. GitHub, [b.r.].

33. CREATION, G-Truc. *GLM* [https://github.com/g-truc/glm]. GitHub, [b.r.].

34. LIBRARY, Open Asset Import. *Assimp* [https://github.com/assimp/assimp]. GitHub, [b.r.].

35. DROLON, Hervé; BERG, Floris van den. *FreeImage* [https://freeimage.sourceforge.io/]. SourceForge, [b.r.].

36. INTEL. *Embree* [https://github.com/embree/embree]. GitHub, [b.r.].

37. MELMAN, Gabi. *spdlog* [https://github.com/gabime/spdlog]. GitHub, [b.r.].

38. MEISTER, Daniel. *Bounding Volume Hierarchies for High-Performance Ray Tracing*. Czech Technical University in Prague. Computing a Information Centre, 2018. Dostupné také z: http://hdl.handle.net/10467/78412. doctoral thesis. Czech Technical University in Prague.

39. KARRAS, Tero. [B.r.]. Dostupné také z: https://developer.nvidia.com/blog/thinking-parallel-part-ii-tree-traversal-gpu/.

40. MACDONALD, David J.; BOOTH, Kellogg S. Heuristics for Ray Tracing Using Space Subdivision. *Vis. Comput.* 1990-05, roč. 6, č. 3, s. 153–166. ISSN 0178-2789. Dostupné z DOI: 10.1007/BF01911006.

41. PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg. 13 - Monte Carlo Integration. In: PHARR, Matt; JAKOB, Wenzel; HUMPHREYS, Greg (ed.). *Physically Based Rendering (Third Edition)*. Third Edition. Boston: Morgan Kaufmann, 2017, s. 747–802. ISBN 978-0-12-800645-0. Dostupné z DOI: https://doi.org/10.1016/B978-0-12-800645-0.50013-0.

42. KŘIVÁNEK, Jaroslav. Image-Based Lighting on Surfaces with Arbitrary BRDF: An Exercise in BRDF Importance Sampling, Sampling from a 2D Discrete Probability Function, and Multiple Importance Sampling. [B.r.]. Dostupné také z: https://cgg.mff.cuni.cz/~pepca/lectures/pdf/pg3-03-ibl-writeup.pdf.

43. GUIBAS, L.; VEACH, Eric. Robust Monte Carlo methods for light transport simulation. In: 1997.

44. KONDAPANENI, Ivo; VEVODA, Petr; GRITTMANN, Pascal; SKŘIVAN, Tomáš; SLUSALLEK, Philipp; KŘIVÁNEK, Jaroslav. Optimal multiple importance sampling. *ACM Transactions on Graphics*. 2019-07-12, roč. 38, č. 4, s. 1–14. ISSN 0730-0301. Dostupné z DOI: 10.1145/3306346.3323009.

Příloha A

Celkové srovnání všech BRDF funkcí

BRDF Funkce	Absolutní čas	Relativní čas
Mirror	0,04 s	1,00
Lambert	0,05 s	1,27
Phong	0,41 s	10,84
Physically correct Phong	0,48 s	12,79
Blinn-Phong	0,50 s	13,25
Cook-Torrance	0,53 s	13,98
Torrance-Sparrow	0,54 s	14,18
Oren-Nayar	1,36 s	36,00

Tabulka A.1: Celkové srovnání všech BRDF funkcí